Politecnico di Torino

Dipartimento di Ingegneria Meccanica e Aerospaziale DIMEAS

Corso di Laurea Magistrale in Ingegneria Meccanica

A.A. 2023/2024

Machine learning per la previsione della risposta a compressione di celle lattice prodotte in Additive Manufacturing

Relatore:
Prof. Andrea Tridello

Candidato:
Giuseppe Memmola

Co-relatori:
Prof. Alberto Ciampaglia
Prof. Carlo Boursier Niutta
Prof. Davide Salvatore Paolino

Abstract

Questo lavoro di tesi tratta della realizzazione di un algoritmo di Machine Learning (ML) in grado di prevedere la risposta a compressione di strutture lattice in lega d'alluminio AlSi10Mg prodotte mediante fabbricazione additiva.

Le scansioni Micro-CT condotte su un campione lattice hanno evidenziato la presenza di porosità interne e di elevata rugosità superficiale. Per considerare queste difettosità è stato adottato un approccio multiscala, partendo dalle analisi agli elementi finiti (FEA) per piccoli volumi rappresentativi (RVE) della cella lattice intera.

Tramite il software Abaqus CAE, è stato realizzato il modello di RVE basandosi sulle piccole travi cilindriche (*beam*) che compongono la struttura reticolare. Nel modello numerico, la difettosità interna è stata considerata utilizzando dei vuoti a forma di ellissoide, mentre la rugosità superficiale è stata simulata con un profilo sinusoidale.

Quindi, sono state eseguite diverse simulazioni FEA sul modello variando i parametri principali di difettosità, al fine di ottenere un database di curve omogeneizzate tensione-deformazione, differenti a seconda degli input utilizzati.

Tali risultati sono poi stati utilizzati per l'addestramento di una rete neurale, la quale, assegnando in input i parametri di difettosità interna e superficiale di una certa beam, possa fornire in output la curva stress-deformazione della beam stessa, sostituendo il processo di analisi agli elementi finiti mediante Abaqus.

In seguito, sul campione lattice sono stati eseguiti test di compressione quasi-statica, al fine di ottenere le curve forza-spostamento sperimentali delle strutture reticolari.

Dunque, con il software LS-Dyna sono state effettuate le simulazioni sulla cella lattice intera (macroscala), inserendo casualmente le diverse varianti di RVE all'interno del componente macroscala. Sono stati svolti due diversi gruppi di simulazioni, associando prima le caratteristiche meccaniche del microscala ottenute mediante le simulazioni su Abaqus, e poi quelle predette dal modello di Machine Learning precedentemente addestrato, ottenendo per entrambi i casi un certo intervallo di variabilità dei risultati.

Infine, con i fasci di curve generate dalle analisi multiscala, sono state determinate due bande di dispersione che, contenendo le curve sperimentali, hanno validato sia le analisi effettuate mediante il software Abaqus, sia i risultati ottenuti con il modello ML.

Indice

Indice delle figure

Indice delle tabelle

1. Introduzione

1.1 Strutture lattice

Le tecnologie di Fabbricazione Additiva, in continua evoluzione negli ultimi anni, consentono la creazione di geometrie e forme complesse che non sono riproducibili utilizzando tecniche di lavorazione meccanica tradizionali. Le strutture lattice sono un importante esempio di ciò che è possibile realizzare grazie alle tecniche di Additive Manufacturing, e, in particolare, la tecnologia di Selective Laser Melting (SLM) ha reso possibile lo sviluppo di queste strutture reticolari che, mediante il controllo di vari parametri, consentono il raggiungimento di proprietà meccaniche, elettriche, termiche e acustiche uniche. Infatti, il potenziale offerto da questo tipo di componente ha fatto sì che questo ricevesse sempre più attenzione, in particolare nei settori biomedico, aerospaziale, automotive e nell'ambito della ricerca.

Le strutture lattice sono caratterizzate da una struttura porosa e ripetitiva delle sue celle unitarie, fino alla formazione del componente completo (Figura 1.1 [1]). Queste strutture possono assumere varie topologie cellulari come celle cubiche, schemi esagonali o strutture più complesse. La scelta della configurazione dipende dall'applicazione specifica e dai requisiti funzionali dell'oggetto finale.

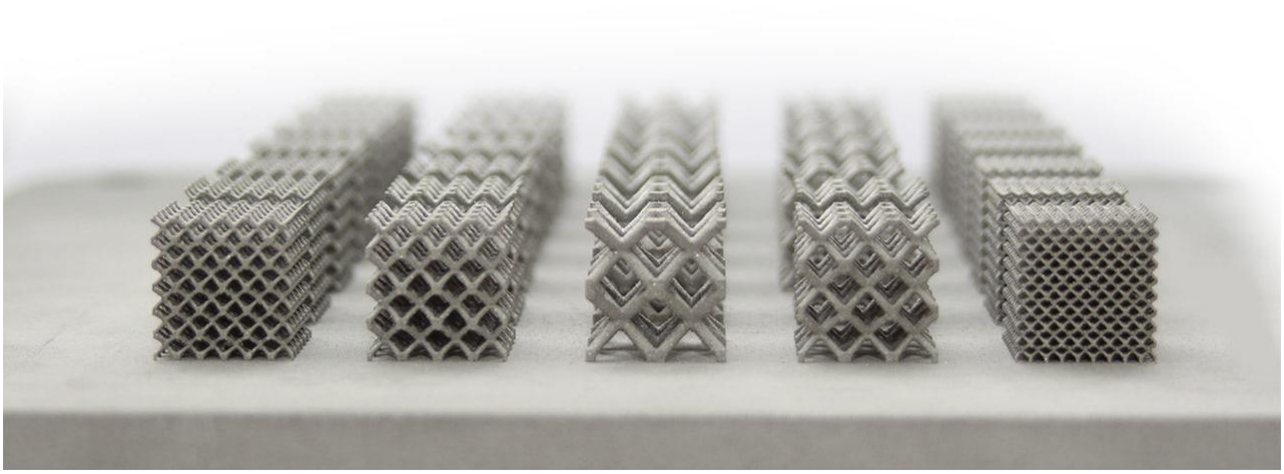

Figura 1. 1 - Esempi di strutture lattice prodotte mediante Fabbricazione Additiva [1]

L'utilizzo di strutture lattice offre diversi vantaggi in vari ambiti. In primo luogo, l'elevata porosità di queste strutture consente una significativa riduzione del peso complessivo dell'oggetto, il che può essere vantaggioso in settori come l'aerospace e l'automotive, dove il peso è un fattore critico. Inoltre, le strutture lattice mostrano elevate resistenza e capacità di assorbimento specifico di energia, nonostante la loro natura leggera; infatti, la distribuzione delle forze attraverso la struttura reticolare consente un'eccellente resistenza meccanica, garantendo la capacità di sopportare carichi significativi. Infine, queste strutture offrono anche una vasta libertà di progettazione, in quanto i vantaggi forniti dall'Additive Manufacturing consentono la produzione di forme complesse e, quindi, componenti ottimizzati per requisiti specifici.

Tuttavia, il problema principale di queste strutture risiede nella presenza di difetti interni e superficiali, da imputare al processo di fabbricazione additiva, che influiscono notevolmente sulle proprietà meccaniche dei componenti. Infatti, ad esempio, "l'elevata concentrazione di intagli, unita alla rugosità superficiale tipica delle condizioni as-built della AM, rende le strutture a reticolo molto vulnerabili ai cedimenti per fatica." [2]

Inoltre, a causa della complessa geometria e del peso computazionale relativo ai modelli numerici, prevedere la risposta sotto sforzo di queste strutture risulta particolarmente difficile, rendendo necessarie costose scansioni Micro-CT per modellazione e simulazione del componente.

Per tale motivo un approccio multiscala e l'utilizzo di un algoritmo di Machine Learning può essere fondamentale per semplificare notevolmente lo studio e l'analisi di queste strutture.

1.2 Approccio multiscala

L'Analisi agli Elementi Finiti (FEA) utilizzata per studiare il comportamento strutturale di componenti meccanici può portare a modelli con elevati costi computazionali quando si analizzano elementi complessi, come le strutture lattice prese in esame. L'uso di un approccio multiscala nell'analisi FEA consente di risolvere questo problema semplificando le simulazioni da effettuare.

Infatti, la modellazione multiscala riguarda la derivazione di equazioni, parametri o algoritmi di simulazione che descrivono il comportamento a una data scala di lunghezza sulla base della fisica a una scala più fine, a condizione che la fisica e la struttura a scala fine siano meglio comprese rispetto a quelle a scala più grossolana [3].

Quindi, questo metodo prevede in un'unica analisi l'integrazione di diverse scale di risoluzione, ciascuna delle quali analizzata con una precisione appropriata, permettendo una maggiore efficienza computazionale senza compromettere l'accuratezza dei risultati.

In primo luogo, le proprietà efficaci vengono calcolate risolvendo un problema a scala fine sul dominio dell'RVE ('Elemento di Volume Rappresentativo') o della cella unitaria, che è definito essere di dimensioni sufficientemente grandi da essere statisticamente rappresentativo del mezzo eterogeneo [3]. Dopo aver definito il micro-volume rappresentativo con le stesse proprietà del componente macro, ad esso si applicano delle condizioni al contorno localizzate.

Le proprietà meccaniche ottenute a livello locale vengono poi trasferite al modello globale mediante tecniche di omogeneizzazione, fornendo una rappresentazione accurata del comportamento complessivo del componente.

In Figura 1.2 è presente un esempio schematico del metodo di progettazione multiscala. Si può evincere come la macrostruttura trasmette lo stato di deformazione di un punto di integrazione al modello microscala e lo applica come condizione al contorno della simulazione di quell’elemento di volume rappresentativo (RVE). I risultati della simulazione in micro vengono quindi utilizzati per aggiornare il livello macro e si costituisce un collegamento tra i livelli macro e micro mediante un processo di omogeneizzazione. In tale processo, i comportamenti microscopici del materiale vengono utilizzati per informare la degradazione, il danno e la possibile rottura al livello della macroscala [4].

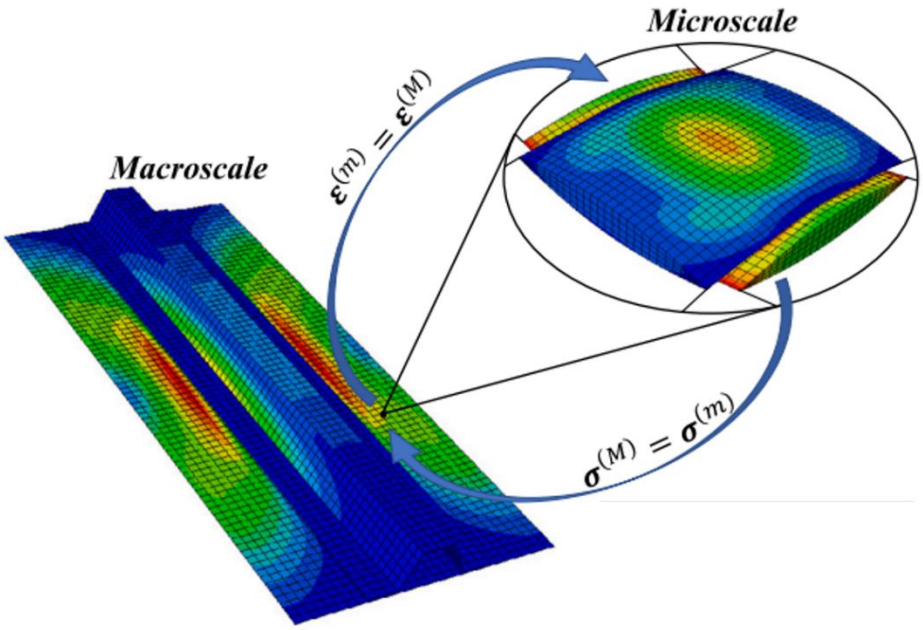

Figura 1. 2 - Esempio schematico di progettazione multiscala [4]

Utilizzare il metodo multiscala nell'analisi FEA di componenti complessi presenta numerosi vantaggi. Innanzitutto, riduce significativamente la complessità computazionale, rendendo possibile l'analisi di componenti che altrimenti richiederebbero tempi di calcolo proibitivi. Inoltre, offre una rappresentazione precisa del comportamento del componente su diverse scale, permettendo una valutazione più approfondita delle proprietà meccaniche dei componenti in esame.

1.3 Machine learning

Il Machine Learning (ML) è un sottoinsieme dell'intelligenza artificiale che si occupa di sviluppare algoritmi e sistemi finalizzati all'apprendimento automatico e al miglioramento delle proprie performance in base ai dati che vengono utilizzati. Esso può rivelarsi un grandissimo alleato dell'approccio multiscala, a causa dell'elevata quantità di dati che possono essere estratti dalle numerose simulazioni che devono essere effettuate in scala locale. Inoltre, poiché eseguire analisi sperimentali e numeriche su geometrie complesse (come le lattice) può diventare piuttosto lungo e costoso, l'utilizzo di algoritmi ML è senza dubbio un approccio da tenere in considerazione.

In una rete neurale di machine learning, mediante l'assegnazione dei parametri di input (x_i), si ottiene una previsione di determinati elementi di output (y_i). Per fare ciò, è necessario addestrare il modello ML attraverso un database di training, affinché la rete neurale impari come correlare gli input agli output per produrre una previsione soddisfacente. Per farlo, fra input e output sono presenti una serie di strati (*layers*) e nodi (detti anche 'neuroni') interconnessi fra loro, come visibile dalla struttura semplificata di rete neurale in Figura 1.3 [5].

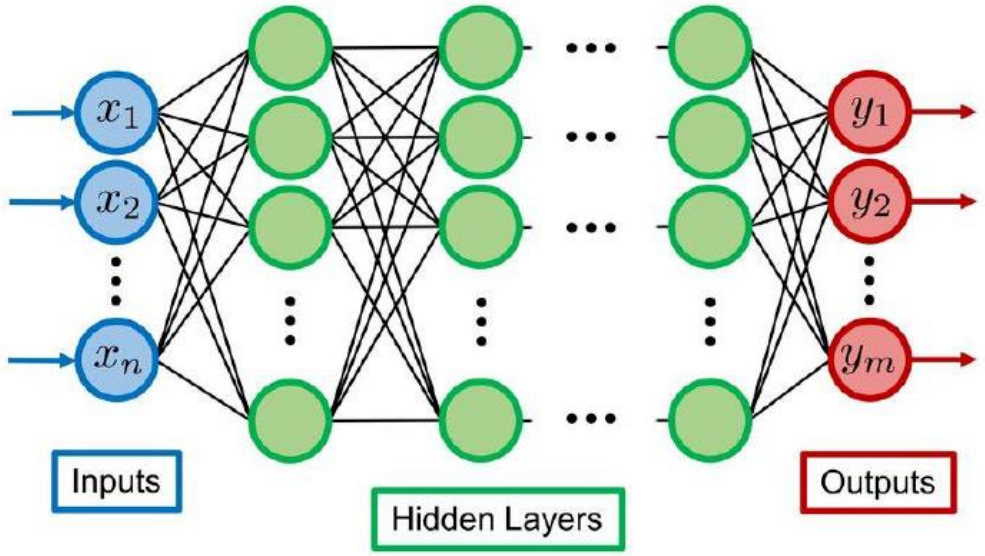

Figura 1. 3 - Struttura di esempio di una rete neurale [5]

Fra gli input e gli output sono presenti gli *hidden layers*, ovvero degli strati intermedi con un elevato numero di nodi al loro interno. Il layer di input invia il segnale al primo 'strato nascosto' mediante la connessione fra i nodi, per poi passare al successivo hidden layer e così via fino al layer di output, da cui si ottiene il risultato finale.

I nodi, connettendosi fra loro, prendono il segnale dall'input e producono un peso w_{ji} (che determina l'importanza della connessione fra i neuroni) e un *bias* b_i (che aiuta a regolare l'output di un neurone); questi, vengono combinati con i valori di input (x_j) per produrre una somma (g_i) calcolata in questo modo [5]:

$$g_i = \sum_j w_{ji} x_j + b_i$$

Successivamente, il valore di g_i viene passato attraverso una particolare funzione che deve essere scelta e specificata per ogni layer, chiamata 'funzione di attivazione' (f_a), mediante la quale si passa ai layer successivi fino a quello di output, ottenendo:

$$y_i = f_{a,i}(g_i)$$

La f_a determina il modo in cui la rete neurale elabora i dati e ce ne possono essere di diverse tipologie, le cui più importanti sono: Lineare, Sigmoid, Rectified Linear Unit (ReLU) e Tangente Iperbolica (tanh).

Un altro elemento importante da scegliere durante l'addestramento della rete neurale è il numero di epoche, in quanto il processo appena descritto viene ripetuto tante volte quanto è il valore di tale parametro. Di conseguenza, man mano che questo processo si ripete, tutti i pesi e i bias vengono regolati ad ogni epoca al fine di aumentare la qualità della previsione. La metrica utilizzata per valutare quanto la previsione sia precisa è detta 'funzione di perdita' (*loss function*) e, solitamente, viene utilizzata quella che minimizza l'errore quadratico medio (M.S.E.) o l'errore assoluto medio (M.A.E.). Quindi, si calcola l'errore per ogni epoca e di conseguenza si aggiornano i pesi e i bias, per assicurare che nella prossima epoca la perdita venga ridotta. Se le epoche sono poche, non si è in grado di addestrare correttamente il modello, poiché il *loss* potrebbe essere troppo alto. Infatti, in generale, è preferibile un alto numero di epoche, ma esagerando potrebbe presentarsi il fenomeno dell'*overfitting*, il quale porta la rete ad allenarsi eccessivamente, non consentendo di ottenere i risultati desiderati.

Un ulteriore parametro molto importante è il tasso di apprendimento (*learning rate*): esso indica alla rete neurale di quanto modificare i bias e i pesi in risposta alla perdita calcolata precedentemente. È fondamentale scegliere correttamente questo parametro, affinché, man mano che si vada avanti col numero di epoche durante l'addestramento, ci sia una costante diminuzione del valore di perdita. Come si può vedere dal grafico in Figura 1.4 ([5]), se il tasso di apprendimento è troppo basso, per ottenere una diminuzione significativa della perdita, è necessario aumentare considerevolmente il numero di epoche; d'altra parte, se il learning rate è troppo alto, la perdita potrebbe aumentare oppure acquisire un comportamento instabile.

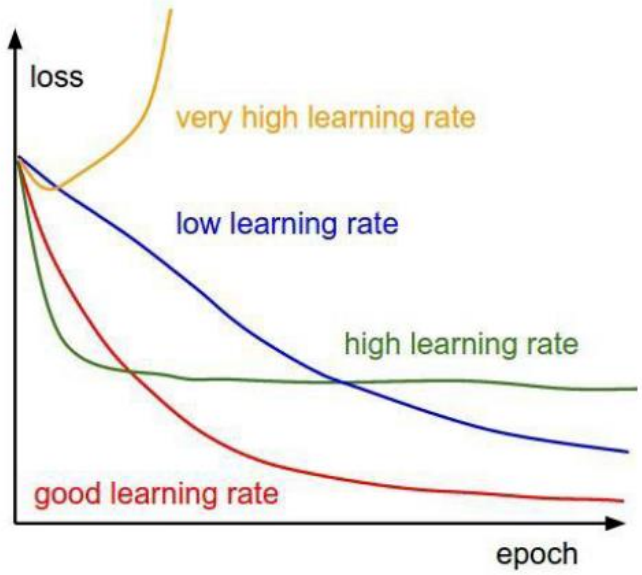

Figura 1. 4 - Comportamento della perdita in funzione del numero di epoche in base al learning rate [5]

Un ultimo parametro da tenere presente è il *batch size*, ovvero il numero di campioni di dati che il modello processa contemporaneamente prima di aggiornare i pesi. Per un batch size piccolo si ha un aggiornamento dei pesi più frequente, il che può rendere l'apprendimento più stabile e permettere al modello di adattarsi meglio alle variazioni dei dati; tuttavia, questo può essere meno efficiente in termini di calcolo e richiedere più tempo per completare un'epoca. Invece, con un batch size elevato si esegue l'aggiornamento dei pesi meno frequentemente, rischiando una convergenza meno stabile, ma che può essere più efficiente dal punto di vista computazionale, sfruttando meglio la parallelizzazione su diversi processori contemporaneamente.

In conclusione, nonostante si possa ricercare il training migliore possibile modificando numero di epoche, tasso di apprendimento e batch size, la qualità dei dati all'interno del database è ciò che determina principalmente se l'addestramento di rete neurale sia efficiente o meno. Per questo motivo, mediante le simulazioni su Abaqus, sarà fondamentale ottenere un set di dati che sia ben rappresentativo dei parametri più influenti sulle proprietà microscala delle celle lattice.

2. Obiettivi del lavoro di tesi

In questo lavoro di tesi si intende analizzare il comportamento meccanico di strutture lattice in AlSi10Mg prodotte tramite fabbricazione additiva. In particolare, l'obiettivo è studiare e prevedere il comportamento di celle lattice sotto carico compressivo quasi statico mediante un approccio multiscala e un modello addestrato di machine learning.

Un aspetto cruciale della ricerca è dato dalla caratterizzazione dei difetti interni e della rugosità superficiale del componente, al fine di ottenere un modello realistico da utilizzare per il metodo multiscala. Scansioni Micro-CT effettuate su campioni di strutture lattice hanno rivelato la presenza di porosità all'interno delle piccole travi della struttura (*beam*), riducendo così il volume effettivo di materiale. Inoltre, l'utilizzo di un profilometro ha permesso l'estrapolazione della rugosità superficiale di un componente as-built prodotto mediante tecnica SLM in AlSi10Mg, preso come campione.

Si parte dalle analisi agli elementi finiti (FEA) e dalla successiva valutazione delle curve tensione-deformazione delle piccole travi (*beam*) che compongono la struttura. L'approccio multiscala permette di considerare la difettosità su scala locale (beam elementare presa come RVE), estraendo le curve di resistenza meccanica utilizzando tecniche di omogeneizzazione, che verranno poi assegnate alla scala macroscopica del campione (struttura lattice). Quindi, si definiscono i parametri che più influenzano le proprietà meccaniche del materiale, e vengono fatti variare al fine di ottenere un elevatissimo numero di curve, differenti a seconda dei parametri utilizzati.

Le curve estratte sono utilizzate per definire un dataset da fornire ad un algoritmo di machine learning, allo scopo di addestrarlo e validarlo per la previsione del comportamento meccanico degli elementi beam in base ai valori dei loro parametri più importanti. In questo modo, fornendo in input le caratteristiche fondamentali di difettosità delle travi elementari, è possibile ottenere in output direttamente il loro comportamento sforzo-deformazione senza dover essere costretti a passare da dispendiose analisi agli elementi finiti, ma semplicemente prevedendone le proprietà con il modello di rete neurale addestrato.

Una volta predetti i comportamenti sforzo-deformazione per gli elementi microscopici compresi di difetti, si passa a valutare l'efficienza del modello ML, assegnando le proprietà predette dalla rete neurale agli elementi che compongono la cella lattice (macroscala).

Infine, sulla struttura vengono eseguite simulazioni FEA per valutarne la risposta a compressione numerica. Dalle analisi si ottiene la risposta forza-spostamento per la struttura reticolare, che viene confrontata con la curva sperimentale per validare il modello ML.

In Figura 2.1 è mostrato il diagramma di flusso che riassume i passaggi seguiti in questo lavoro di tesi, in cui sono evidenziati l'approccio multiscala e l'utilizzo del machine learning.

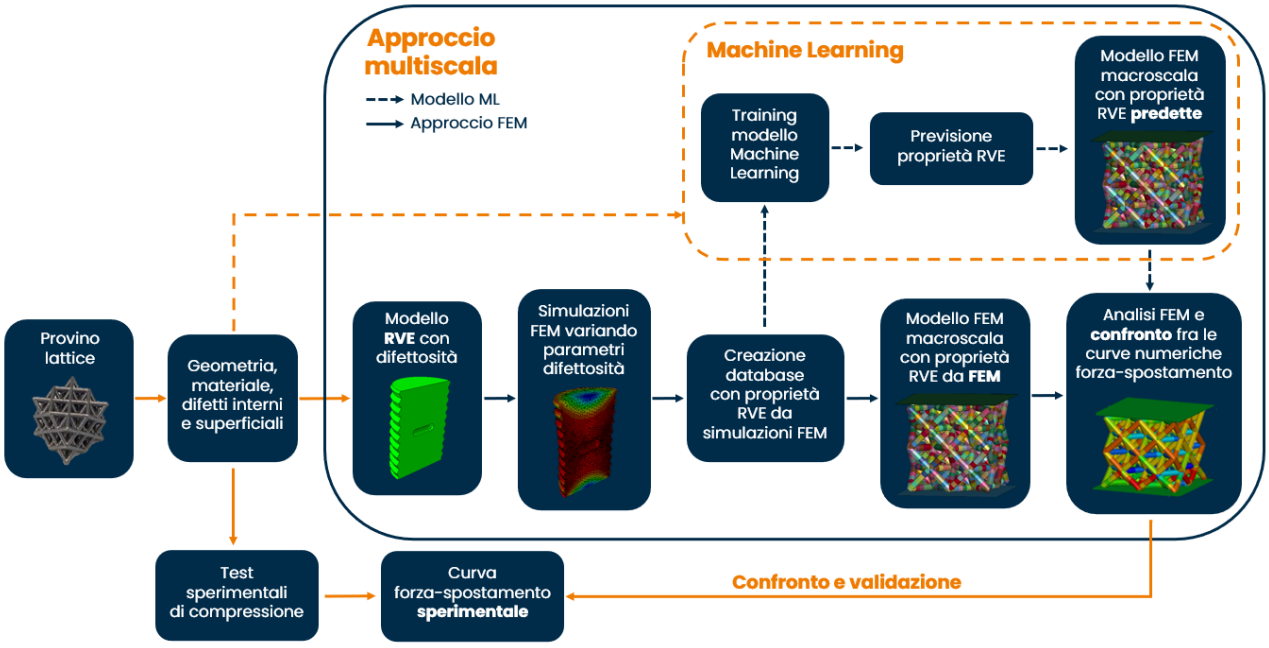

Figura 2. 1 - Diagramma di flusso rappresentativo dei passaggi seguiti utilizzando un approccio multiscala e un modello ML

3. Modello RVE cilindrico per il metodo multiscala

Per l'applicazione di un metodo multiscala per la risoluzione numerica del problema, è necessario dividere lo studio fra un livello microscala e uno macroscala. Si è partiti da un'analisi microscala, per la quale è stato definito un modello di RVE a forma di trave cilindrica (*beam*) che avesse caratteristiche analoghe alla struttura macro. Quest'ultima è rappresentata da una cella lattice prodotta tramite Fabbricazione Additiva, utilizzando la tecnica della fusione laser selettiva (SLM) e impiegando polvere di lega di alluminio AlSi10Mg. In Figura 3.1 è mostrato un esempio di cella lattice molto simile a quella analizzata in questo lavoro di tesi.

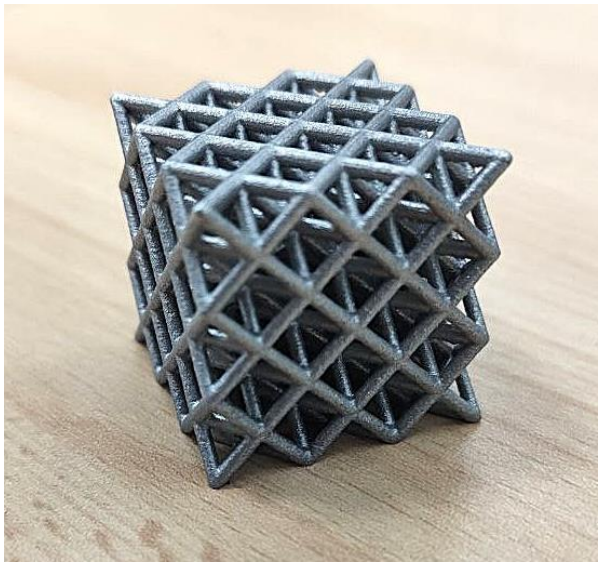

Figura 3. 1 - Esempio di struttura lattice prodotta in AlSi10Mg mediante tecnica SLM

La geometria di un campione lattice può essere ricostruita tridimensionalmente (esempio in Figura 3.2) con delle scansioni mediante microtomografia computerizzata (Micro-CT), al fine di visualizzarne le parti interne e superficiali in modo dettagliato, evidenziando aspetti microscopici che possono non essere visibili in una semplice immagine bidimensionale. I risultati ottenuti da queste scansioni saranno ripresi più volte nelle pagine successive, in quanto fondamentali per le analisi che verranno affrontate in questo studio.

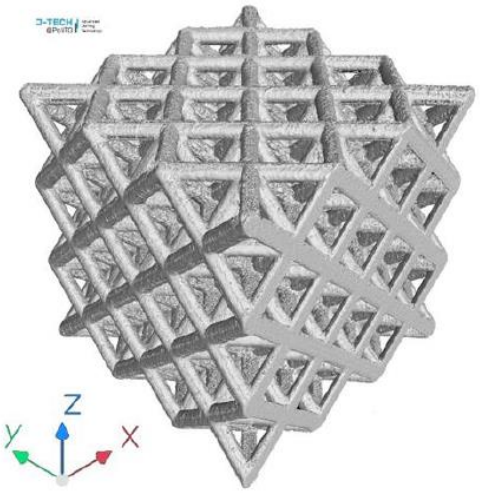

Figura 3. 2 - Ricostruzione tridimensionale di un campione lattice di esempio dopo una scansione Micro-CT

3.1 Geometria RVE e proprietà materiale

Al fine di rappresentare una porzione delle piccole travi che compongono la struttura lattice, la geometria dell'RVE consiste in un cilindro con un diametro di 1.4 mm e un'altezza di 2 mm, dimensioni corrispondenti a quelle con cui la cella lattice presa come campione è stata discretizzata mediante le scansioni Micro-CT (Figura 3.3, ripresa da Abaqus CAE). La rugosità superficiale e i difetti interni del modello cilindrico verranno analizzati più nel dettaglio nei paragrafi successivi.

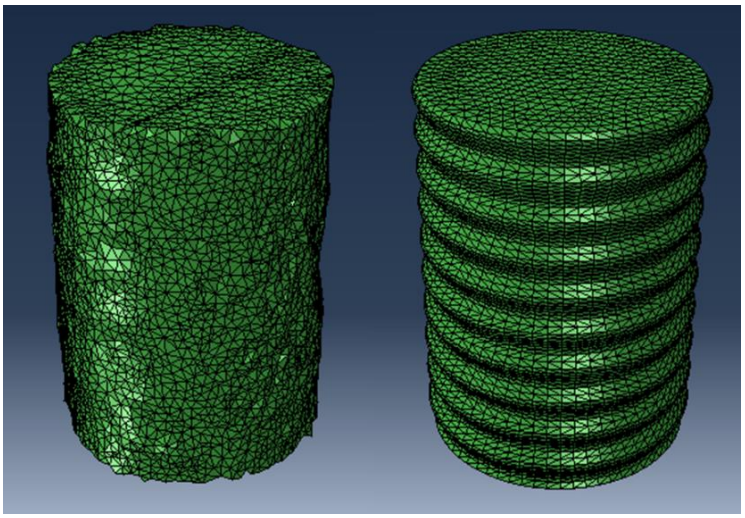

Figura 3. 3 - Beam estratta con Micro-CT (sinistra) e modello di RVE cilindrico (destra)

Da precedenti test di trazione condotti presso il laboratorio DIMEAS del Politecnico di Torino su campioni di AlSi10Mg as-built prodotti mediante SLM, è stata ricavata la curva tensione-deformazione mostrata in Figura 3.4. Poiché le strutture lattice analizzate in questo studio hanno il medesimo materiale e processo di produzione, questi risultati vengono utilizzati per valutare l'influenza della tecnica di Additive Manufacturing sulle proprietà meccaniche del componente.

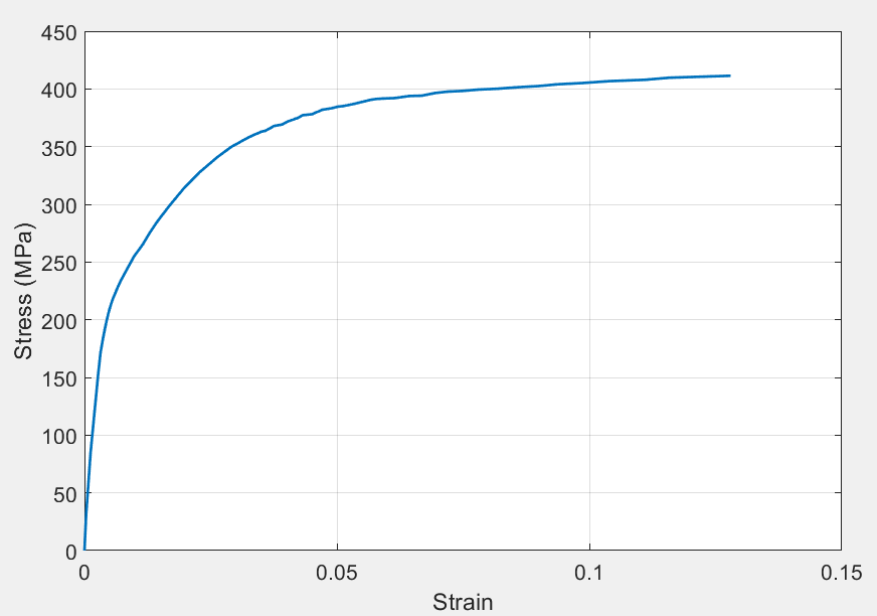

Figura 3. 4 - Curva stress-strain AlSi10Mg in seguito a prova di trazione

Ulteriori test, come quelli di nanoindentazione, necessari per determinare i valori di durezza Vickers e modulo elastico, sono stati condotti direttamente sulla struttura lattice nei laboratori del Politecnico di Torino, ottenendo le proprietà meccaniche elencate in Tabella 3.1.

AlSi10Mg (campione struttura lattice)	
Densità (ρ)	2.7 g/cm^3
Modulo elastico (E)	53 GPa
Coefficiente di Poisson (ν)	0.33

Tabella 3. 1 - Proprietà meccaniche considerate per l'RVE cilindrico

La curva tensione-deformazione e queste caratteristiche del materiale sono state prese in considerazione per le analisi FE che verranno mostrate nelle pagine seguenti.

3.2 Analisi FE

Sul modello di RVE cilindrico sono state effettuate simulazioni FEA (Analisi agli Elementi Finiti) utilizzando il software Abaqus CAE, per determinarne il comportamento omogeneizzato tensione-deformazione. Per eseguire le analisi FE è stato creato uno script Python per modellare l'RVE cilindrico con le sue caratteristiche meccaniche e geometriche, inserendo le condizioni al contorno desiderate e avviando le simulazioni.

3.2.1 Proprietà modello e parametri simulazione

Al modello sono state assegnate la densità e le proprietà elastiche del materiale presenti in Tabella 3.1, mentre come proprietà plastica di riferimento è stata inserita nel modello Abaqus la regione plastica (Tabella 3.2) estratta dalla curva stress-strain del provino di AlSi10Mg (già vista in Figura 3.4).

	Stress [MPa]	Deformazione plastica		Stress [MPa]	Deformazione plastica
1	152,729	0	29	374,993	0,039603
2	171,113	0,000454	30	377,232	0,040481
3	183,939	0,000960	31	378,090	0,042343
4	193,871	0,001410	32	381,984	0,044416
5	205,585	0,002012	33	383,216	0,046203
6	214,580	0,002623	34	384,619	0,047454
7	218,596	0,002957	35	385,113	0,048513
8	228,345	0,003892	36	386,852	0,050480
9	233,700	0,004469	37	388,316	0,051859
10	241,399	0,005403	38	390,521	0,053927
11	255,123	0,007139	39	391,229	0,054908
12	265,501	0,008875	40	391,718	0,056602
13	274,874	0,010143	41	392,030	0,058700
14	283,225	0,011415	42	393,860	0,061636
15	288,463	0,012296	43	394,133	0,064061
16	299,693	0,014282	44	396,447	0,066724
17	313,929	0,016983	45	397,699	0,069339
18	327,744	0,020082	46	398,215	0,072025
19	340,612	0,023527	47	399,430	0,075342
20	349,720	0,026342	48	400,099	0,078853
21	358,335	0,029912	49	401,498	0,083277
22	362,872	0,032264	50	402,493	0,087327
23	363,760	0,033173	51	404,059	0,091137
24	365,222	0,033717	52	404,998	0,095599
25	366,071	0,034113	53	406,721	0,101176
26	367,834	0,034779	54	407,904	0,108304
27	369,184	0,036434	55	409,713	0,113148
28	371,882	0,037625	56	411,392	0,125276

Tabella 3. 2 - Regione plastica estratta dalla curva stress-strain del provino in AlSi10Mg

Per le analisi FE si è scelto di attivare il parametro *nlgeom*, il quale consente ad Abaqus di calcolare gli spostamenti e le deformazioni in base alla configurazione geometrica istante per istante, e non basandosi solo sulla geometria iniziale. In questo modo, vengono garantiti risultati più accurati per tensioni e deformazioni, considerando interazioni non lineari tra i materiali e la geometria.

Inoltre, mediante il codice Python, è stata definita una mesh pari a 0.07 mm come buon compromesso fra precisione da ottenere durante la simulazione e tempo necessario per le analisi. Inoltre, sono state scelte strategicamente alcune zone di raffinamento della mesh nei punti in cui fossero previsti degli stress e delle deformazioni più critiche: in particolare, è stata applicata una mesh più fitta (pari a 0.03 mm) lungo la superficie dell'RVE cilindrico e nella zona centrale del modello, in cui è presente il difetto interno. Un giusto bilanciamento fra mesh predefinita e raffinata è cruciale per ottenere dei buoni risultati senza aumentare eccessivamente il numero di elementi (il risultato finale è visibile in Figura 3.5).

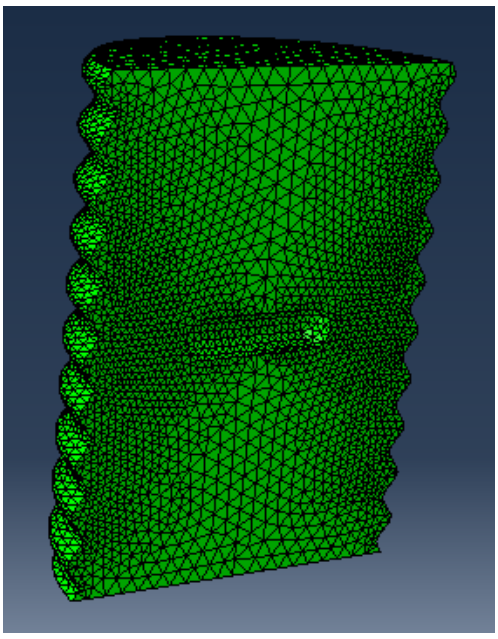

Figura 3. 5 - Mesh assegnata all'RVE

Infine, sono stati definiti due set di nodi, *Top* e *Bottom*, che contengono i nodi alla sommità e alla base del cilindro: questi set verranno utilizzati per applicare le condizioni di carico e di vincolo specifici della simulazione.

3.2.2 Condizioni al contorno

Le condizioni al contorno inserite su Abaqus, visibili in Figura 3.6 e 3.7, simulano una prova di trazione dell'RVE. In particolare, una faccia del cilindro è stata interamente bloccata per quanto riguarda le traslazioni lungo i tre assi, mantenendo però libere le rotazioni, affinché il diametro della beam si potesse restringere in seguito all'allungamento dovuto alla prova di trazione. Sulla faccia opposta è stato applicato uno spostamento di trazione pari a u2 = 0.2 mm lungo l'asse y, mantenendo bloccate le traslazioni lungo gli altri due assi e permettendo la rotazione come nel caso delle condizioni di vincolo.

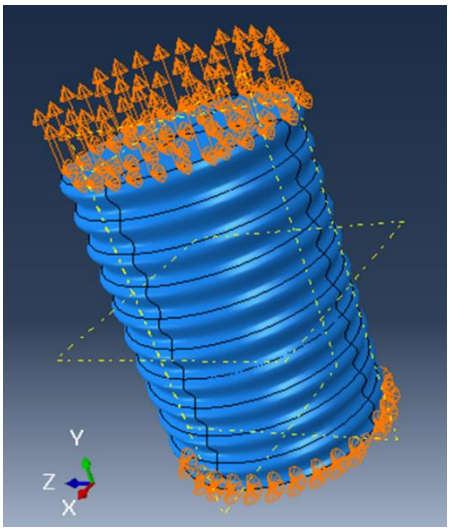

Figura 3. 6 – Rappresentazione grafica su Abaqus delle condizioni al contorno per il modello

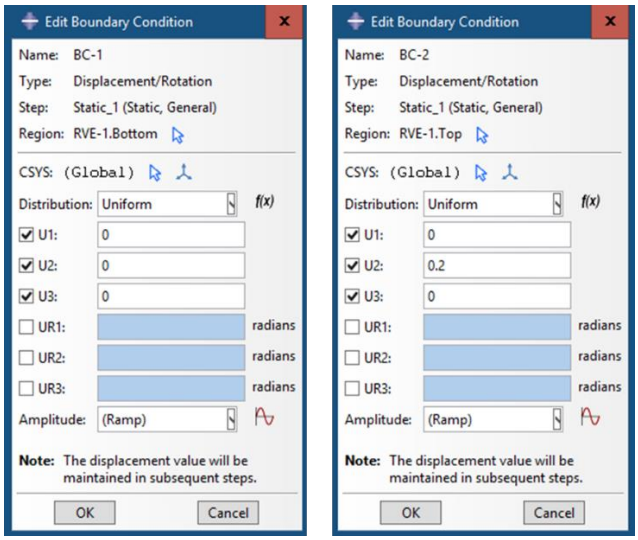

Figura 3. 7 - Condizioni di vincolo (sinistra) e di carico (destra) per il modello Abaqus

3.2.3 Curva omogeneizzata

Per stabilire le curve Stress-Strain dell'RVE è necessario determinare i valori di sforzo e deformazione per ogni frame di output della simulazione corrente. Le componenti di tensione e deformazione scelte per queste operazioni sono rispettivamente lo stress equivalente di Von Mises e la deformazione massima principale. In Figura 3.8, ripresa dal file *.odb* di output per le simulazioni su Abaqus, è presente un esempio di rappresentazione grafica di queste due componenti per l'RVE utilizzato in questo studio.

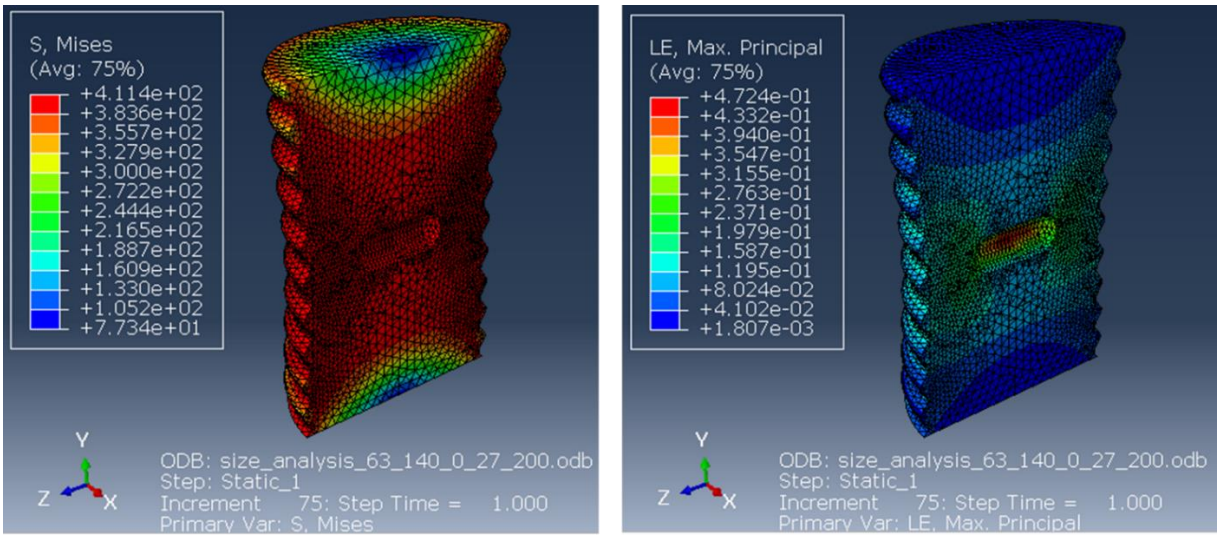

Figura 3. 8 - Stress di Von Mises (sinistra) e deformazione massima principale (destra) per un RVE simulato

È stato quindi scritto un codice Python che, una volta terminata l'analisi FE, legga i file *.odb* di output, estragga per ogni frame la tensione e la deformazione applicata a ciascun volume della mesh e calcoli una media ponderata su tutti gli elementi per ottenere un valore unico per l'intero RVE, secondo le formule discrete:

$$\sigma_{medio} = \frac{\sum_i \sigma_i V_i}{V_{tot}} \quad , \quad \varepsilon_{medio} = \frac{\sum_i \varepsilon_i V_i}{V_{tot}}$$

In questo modo, per ogni simulazione, si ottiene una curva omogeneizzata che rappresenti il comportamento della beam, considerando il contributo medio delle sue proprietà microscopiche e, quindi, degli elementi costituenti la mesh.

Si è scelto di suddividere la simulazione in 75 frame di output, al fine di ottenere 75 punti circa equidistanti per la definizione della curva omogeneizzata. Questa scelta è stata fatta per rappresentare adeguatamente sia il campo elastico che quello plastico con un numero sufficiente di punti e, in particolare, per ottenere un punto di fine tratto elastico abbastanza accurato, essendo uno dei valori basilari di una curva stress-deformazione. Tale punto è fondamentale per questo lavoro di tesi, in quanto necessario per la ricostruzione della curva tensione-deformazione dopo l'utilizzo del modello di machine learning (come verrà spiegato approfonditamente nel capitolo dedicato).

Si è stabilito arbitrariamente che questo punto di fine tratto elastico venga definito come il primo per cui ci sia una variazione di inclinazione della curva maggiore del 5% rispetto all'inclinazione iniziale, dove ogni inclinazione è data dal rapporto fra lo stress e la deformazione in ciascun punto ed è pari al modulo di Young ($E_i = \sigma_i / \varepsilon_i$). Infatti, la rigidezza E della beam sarà definita come quella ricavata nel punto di fine tratto elastico, differente per ciascuna curva ottenuta dalle analisi.

Infine, è stato determinato un metodo per definire univocamente il punto di rottura, ovvero l'altro punto chiave della curva omogeneizzata. Non tutte le parti del modello cederanno contemporaneamente, ma alcuni elementi più sollecitati raggiungeranno il valore di deformazione massima prima degli altri. Nella generazione delle curve omogeneizzate, questa variabilità è stata presa in considerazione tenendo conto del numero di elementi che superino il valore di deformazione massima del materiale per ogni frame di risposta. La generazione della curva, quindi, si interrompe nel frame in cui almeno il 10% degli elementi della mesh supera localmente il valore di *ultimate strain*, determinando la rottura della beam. È stato scelto di considerare una certa percentuale di elementi che raggiungano rottura in quanto, a causa della presenza di mesh raffinata all'interno del modello, potrebbero presentarsi elementi distorti all'interno dell'RVE, che potrebbero sviluppare concentrazioni di sforzo e valori di deformazione eccessivamente alti.

Le curve omogeneizzate così definite saranno quindi fondamentali in questo studio, sia come parametro di confronto fra le diverse casistiche che verranno analizzate, e sia per definire le caratteristiche meccaniche della struttura lattice intera (macroscala) nelle simulazioni agli elementi finiti della stessa.

3.3 Rugosità superficiale

Mediante un microscopio elettronico a scansione (FESEM) sono state estratte immagini ravvicinate di una cella lattice (Figura 3.9), mostrando la presenza di elevata rugosità superficiale, una delle principali problematiche legate ai componenti prodotti in Additive Manufacturing. Questa caratteristica è particolarmente accentuata nelle strutture lattice, in cui la complessa geometria rende molto complicate le operazioni di finitura superficiale.

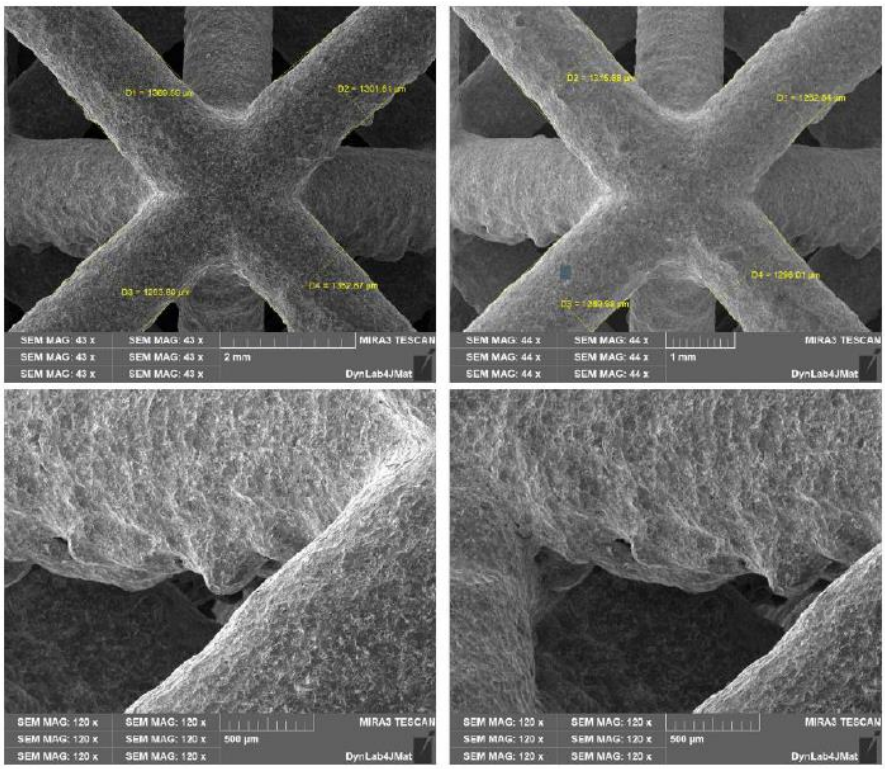

Figura 3. 9 – Rugosità superficiale di una cella lattice ripresa dal microscopio elettronico a scansione

Con l'utilizzo di un rugosimetro è stata estratta la profilometria di un pezzo as-built prodotto in AlSi10Mg mediante SLM (Figura 3.10), ottenendo dei parametri di rugosità realistici (Tabella 3.3) che possono essere utilizzati per simulare al meglio la difettosità superficiale della beam.

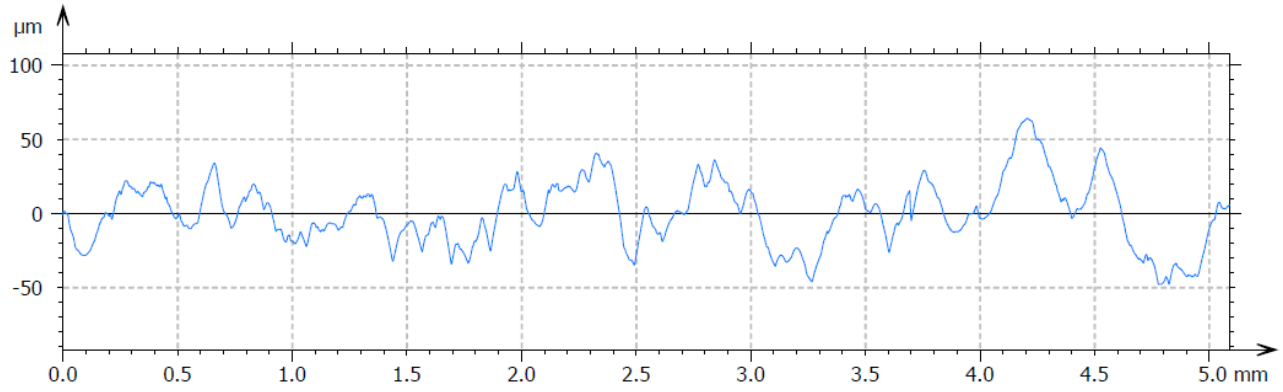

Figura 3. 10 - Profilometria di componente as-built in AlSi10Mg

ISO 4287			
Amplitude parameters - Rough...			
Rp	27.9	µm	Gaussian filte...
Rv	27.6	µm	Gaussian filte...
Rz	55.4	µm	Gaussian filte...
Rc	38.5	µm	Gaussian filte...
Rt	72.9	µm	Gaussian filte...
Ra	11.6	µm	Gaussian filte...
Rq	13.8	µm	Gaussian filte...
Rsk	0.000734		Gaussian filte...
Rku	2.19		Gaussian filte...
Material ratio parameters - Rou...			
Rmr	0.385	%	c = 1 µm und...
Rdc	27.3	µm	p = 20%, q...

Tabella 3. 3 - Parametri di rugosità per componente as-built in AlSi10Mg

3.3.1 Approssimazione con sinusoide

La rugosità superficiale del modello è stata simulata come variazione di spessore della trave per mezzo di un profilo sinusoidale, analogamente ad un metodo presente in letteratura (Figura 3.11 [6]).

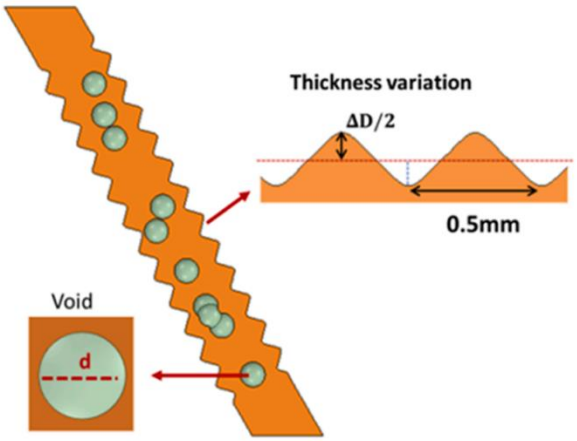

Figura 3. 11 - Rugosità superficiale simulata come variazione di spessore [6]

Quindi, l'RVE cilindrico è stato modellato affinché il suo diametro variasse sinusoidalmente con un'ampiezza pari a 27.75 µm, media fra Rp (altezza massima di picco del profilo) e Rv (profondità massima di valle) estratti col rugosimetro (Tabella 3.3), e periodo pari a 200µm, al fine di ottenere un profilo sinusoidale completo lungo tutti i 2 mm di lunghezza della beam. Il profilo ottenuto è una buona approssimazione di quello estratto mediante la profilometria, come si può evincere dal confronto fra i due profili su un campione di 2 mm, presente in Figura 3.12.

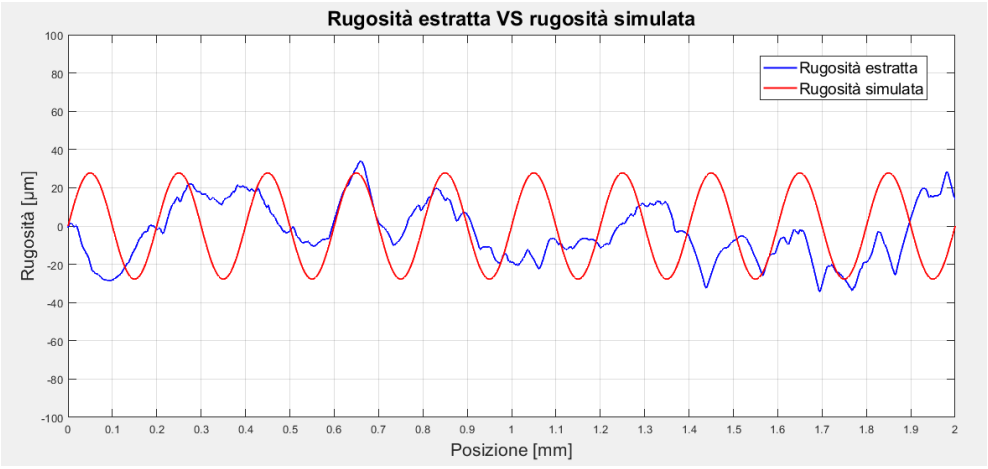

Figura 3. 12 - Confronto fra profilo estratto e simulato

3.3.2 Confronto fra modello RVE e Micro-CT

Per confermare che tale profilo simulasse bene la rugosità superficiale reale della beam, mediante Abaqus sono state eseguite analisi FE di trazione per le due configurazioni (Figura 3.13), entrambe per il momento prive di difetto interno al fine di considerare unicamente l'influenza della rugosità superficiale.

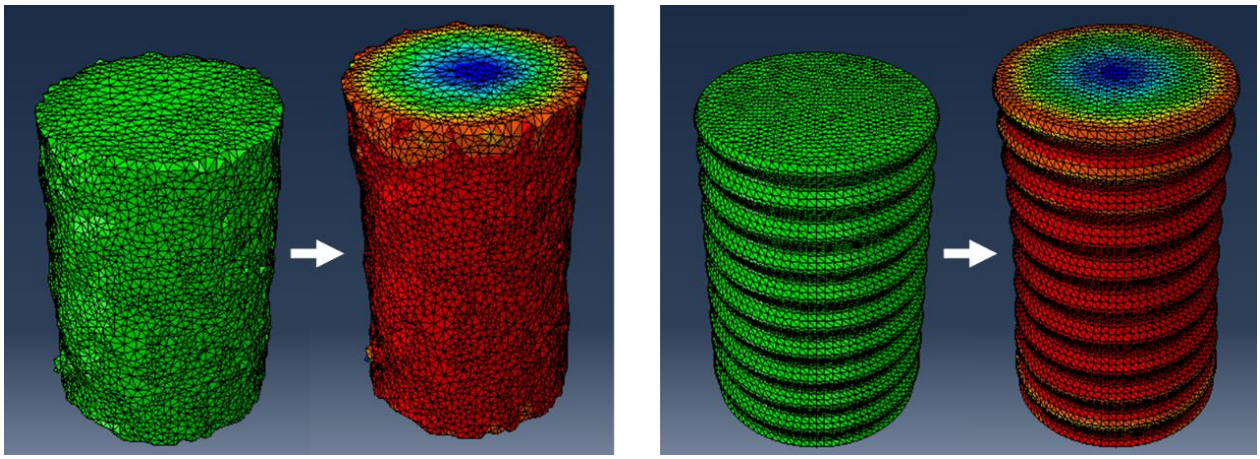

Figura 3. 13 - Analisi FE di trazione mediante Abaqus per la beam estratta con Micro-CT (sinistra) e il modello RVE (destra)

Dalle curve omogeneizzate estratte (Figura 3.14) si evince una certa similitudine fra le due casistiche. In particolare, si ha un basso valore di errore quadratico medio (RMSE = 2.22) fra i due andamenti, con le proprietà meccaniche del caso di rugosità simulata leggermente inferiori rispetto al caso di beam con superficie reale, come si può vedere in Tabella 3.4. Questo non risulta essere un problema in quanto ci pone su un piano di maggiore sicurezza per la valutazione del comportamento omogeneizzato dell'RVE cilindrico.

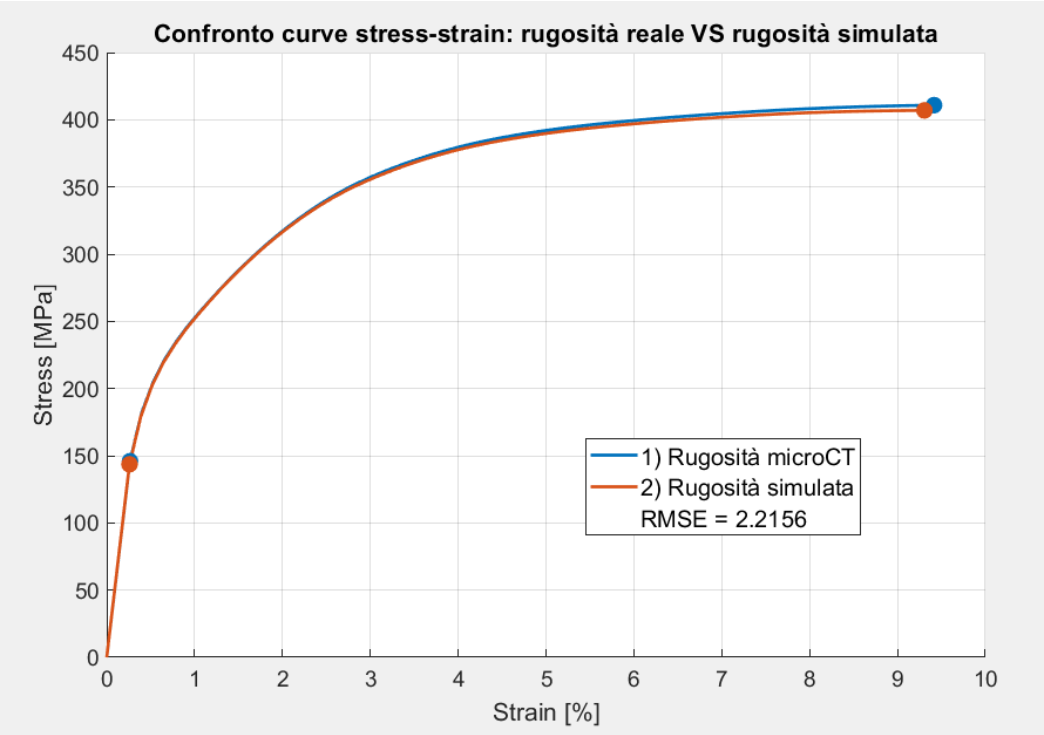

Figura 3. 14 - Confronto curve omogeneizzate: rugosità reale VS rugosità simulata

Beam 1.4mm x 2mm NO difetto interno	Rugosità Micro-CT	Rugosità simulata	Var. %
E [GPa]	55,211	55,380	0,306
UTS [MPa]	410,86	406,97	-0,947
ε_max [%]	9,42	9,31	-1,168

Tabella 3. 4 - Confronto proprietà meccaniche: rugosità reale VS rugosità simulata

3.4 Difetto interno

3.4.1 Scansioni Micro-CT per difetti interni

Mediante la scansione Micro-CT di un campione lattice 2x2 (ovvero con dimensioni 20 x 20 x 20 mm) è stata rilevata la presenza di difetti interni dovuti al processo di Fabbricazione Additiva, che diminuiscono il volume effettivo del materiale e, di conseguenza, la sua resistenza meccanica. Attraverso le scansioni mediante microtomografia computerizzata in vari punti della struttura, le porosità evidenziate mostrano una geometria che può essere approssimata come sferica o, più in generale, ellissoidale. In Figura 3.15 è mostrata la vista sul piano XY di una scansione, evidenziando la presenza di un difetto in un nodo della struttura.

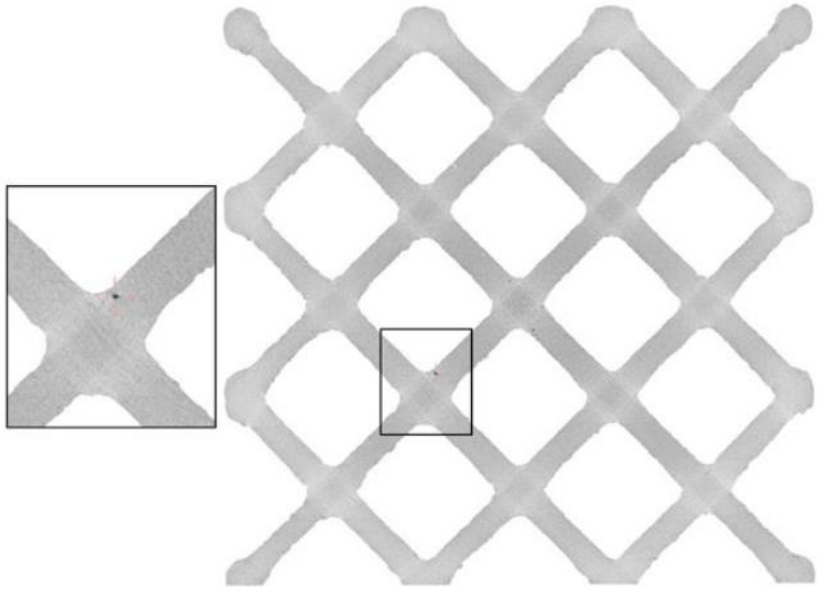

Figura 3. 15 - Vista sul piano XY di una scansione con evidenziato un difetto interno in un nodo fra due beam

I dati relativi a tutti i possibili difetti interni, caratterizzati dai loro diametri massimi, sono stati raccolti e riportati in un documento del software Microsoft Excel. Sono state definite 21 classi di difetto basandosi sui valori di diametro massimo estratti dalle scansioni Micro-CT: in Tabella 3.5 sono presenti le classi di difetto con cui la distribuzione è stata discretizzata, con i corrispondenti valori di diametro massimo e frequenza.

Cella 2x2		
ID	D_{max} [mm]	Frequenza
1	0.062	0.0951
2	0.087	0.2597
3	0.111	0.2452
4	0.136	0.1254
5	0.161	0.0904
6	0.186	0.0782
7	0.21	0.0536
8	0.235	0.0308
9	0.26	0.0111
10	0.284	0.00367
11	0.309	0.00242
12	0.334	0.00137
13	0.359	0.00125
14	0.383	0.0006
15	0.408	0.00044
16	0.433	0.00028
17	0.458	0.00012
18	0.482	0.00008
19	0.532	0.00008
20	0.556	0.00004
21	0.631	0.00004

Tabella 3. 5 - Classi di difetto interno per cella 2x2 con corrispondenti diametro massimo e frequenza

È stata definita la frequenza di ciascuna classe analizzando quante volte ciascuna dimensione si ripetesse all'interno della cella lattice. In Figura 3.16 è presente l'andamento della frequenza dei difetti per la struttura 2x2, da cui si può notare una certa concentrazione della distribuzione intorno ad un diametro massimo pari a 0.1 mm.

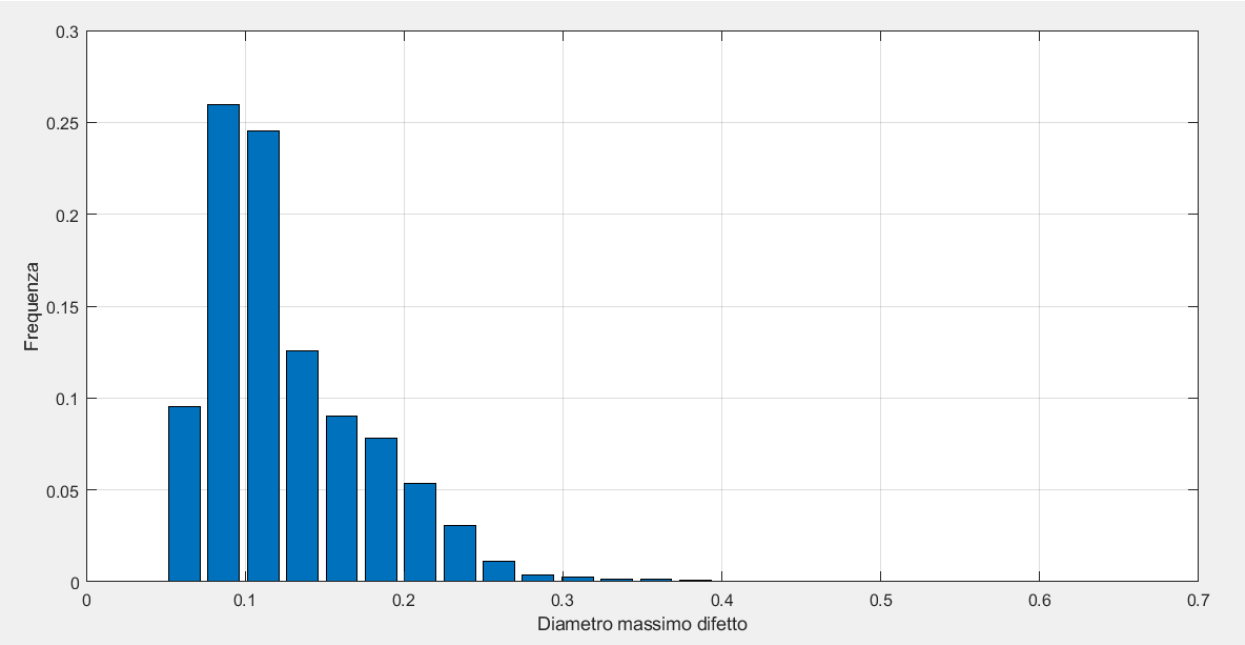

Figura 3. 16 – Frequenza delle dimensioni dei difetti nel campione 2x2

Per questo lavoro di tesi, ottenute le classi di difetto dai campioni sperimentali, i risultati riguardanti la cella 2x2 verranno impiegati per definizione e validazione del modello numerico utilizzato per l'approccio multiscala.

3.4.2 Forma difetto interno modello RVE

Come anticipato nel paragrafo precedente, la forma del difetto può essere assunta generalmente come ellissoidale, quindi con tre semi-assi di diverse dimensioni.

La sezione di modeling del software Abaqus, però, non dispone di una funzione per la creazione di ellissoidi con tre semi-assi differenti e, quindi, è stato necessario ideare una geometria che potesse approssimare bene una forma ellissoidale per il difetto interno. Si è optato per la geometria rappresentata in Figura 3.17, costituita da un cilindro a base ellittica, a cui vengono aggiunti alle due estremità due semi-ellissoidi.

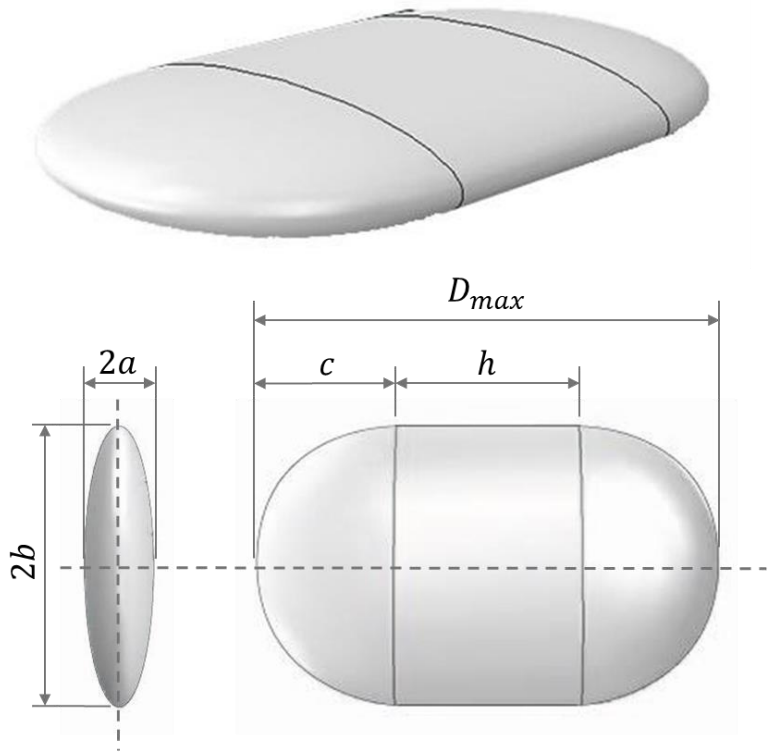

Figura 3. 17 - Forma (in alto) e quote (in basso) del difetto interno considerato per il modello di RVE

In particolare, il volume del difetto sarà dato dalla formula:

$$V = V_{ellissoide} + V_{cilindro} = \frac{4}{3}\pi\, a\, b\, c + \pi\, a\, b\, h$$

I valori delle quote sono invece dati da:

$$2a = \frac{D_{min}}{AR}$$

$$2b = D_{min} \cdot AR$$

$$c = \frac{D_{min}}{2} \cdot AR$$

$$h = D_{max} - (D_{min} \cdot AR)$$

Si può notare, quindi, come la geometria del difetto dipenda da tre parametri principali: diametro massimo (D_{max}), diametro minimo (D_{min}) e *aspect ratio* (AR).

L'aspect ratio è un parametro che descrive la proporzione tra la larghezza e l'altezza di un elemento. In questo caso di studio, l'AR caratterizza la forma del difetto ellissoidale, essendo considerato come il rapporto fra i due semiassi minori del difetto stesso. In Figura 3.18 sono presenti diversi valori di AR per un difetto ellissoidale [7]. Dall'immagine è possibile notare che nel caso con AR=1 si formi un cerchio come proiezione dell'ellissoide sul piano XY; infatti, nell'ipotesi di ellissoide con tre semi-assi uguali, si otterrebbe un difetto sferico, facendo intuire come una forma ellissoidale per il difetto sia la più generica possibile.

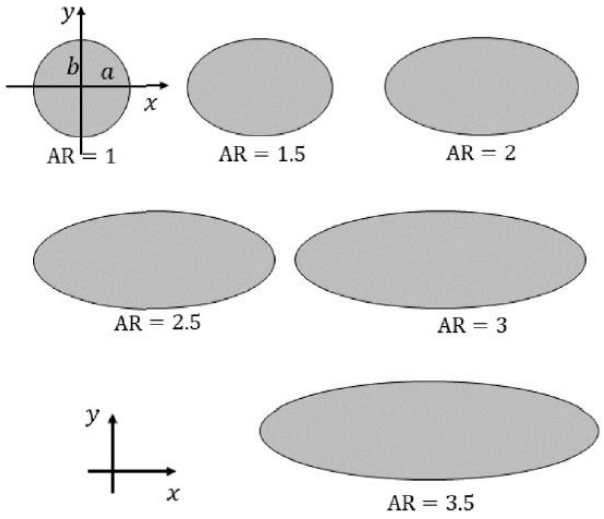

Figura 3. 18 - Difetti ellissoidale con diversi valori di aspect ratio [7]

L'altro parametro fondamentale per la definizione della geometria del difetto è il D_{min}, il quale non è direttamente estraibile dalle scansioni Micro-CT, ma è necessario ricavarlo dai valori di diametro massimo, volume e aspect ratio del difetto interno.

A causa della geometria non perfettamente ellissoidale della porosità interna, è stato necessario definire una equazione che, partendo dalla formula del volume della beam, renda il diametro minimo funzione di D_{max}, V e AR:

$$V = \frac{4}{3}\pi\, a\, b\, c + \pi\, a\, b\, h = \ \ldots\ = \frac{\pi}{4}\cdot D_{max}\cdot {D_{min}}^2 - \frac{\pi}{12}\cdot AR\cdot {D_{min}}^3 \Rightarrow$$

$$\Rightarrow \left(-\frac{\pi}{12}\cdot AR\right)\cdot {D_{min}}^3 + \left(\frac{\pi}{4}\cdot D_{max}\right)\cdot {D_{min}}^2 - V = 0 \quad \Rightarrow$$

$$\Rightarrow \quad D_{min} = f(AR, V, D_{max})$$

Si ottiene, quindi, un'equazione di terzo grado del tipo:

$$\alpha\, x^3 + \beta x^2 + \gamma = 0$$

Risolvendo questa equazione e imponendo che il risultato sia compreso fra 0 e il valore del D_{max} corrispondente alla classe di difetto considerata, si ottiene il valore di D_{min} in funzione di diametro massimo, volume e aspect ratio. In Tabella 3.6 sono riportati i valori ottenuti dal calcolo di D_{min} per tutte le 21 classi di difetto (della cella 2x2) al variare dell'aspect ratio.

ID	D_{max} [mm]	AR = 1	AR = 1.5	AR = 2
		D_{min} [mm]	D_{min} [mm]	D_{min} [mm]
1	0.062	0.037	0.041	0.048
2	0.087	0.042	0.044	0.048
3	0.111	0.045	0.047	0.05
4	0.136	0.049	0.051	0.053
5	0.161	0.063	0.066	0.07
6	0.186	0.075	0.078	0.083
7	0.21	0.081	0.084	0.089
8	0.235	0.083	0.086	0.09
9	0.26	0.088	0.091	0.095
10	0.284	0.102	0.106	0.111
11	0.309	0.108	0.112	0.117
12	0.334	0.115	0.119	0.124
13	0.359	0.112	0.116	0.12
14	0.383	0.118	0.122	0.127
15	0.408	0.114	0.117	0.121
16	0.433	0.117	0.121	0.124
17	0.458	0.132	0.136	0.141
18	0.482	0.116	0.119	0.122
19	0.532	0.133	0.136	0.140
20	0.556	0.134	0.137	0.141
21	0.631	0.163	0.167	0.172

Tabella 3. 6 - Valori di D_{min} al variare dell'aspect ratio per ciascuna classe di difetto della cella 2x2

Si può evincere come siano stati considerati tre precisi valori di AR (pari a 1, 1.5 e 2), corrispondenti a quelli che verranno utilizzati per la creazione del piano sperimentale, descritto nelle pagine successive. Inoltre, non vengono mostrati i valori di volume, ma solo quelli di diametro massimo, in quanto a ciascuna classe corrisponde un unico D_{max} e un unico V, ma la misura del primo è concettualmente più semplice per il confronto fra le diverse dimensioni di difetto.

Con la definizione della forma del difetto, si completa la descrizione del modello di RVE cilindrico di base che verrà utilizzato per l'approccio multiscala. Ora, dunque, è possibile passare alla fase di simulazione su Abaqus per determinare i parametri di difettosità che più influenzano le proprietà meccaniche della beam.

4. Parametri influenti su proprietà RVE

In questo capitolo verranno esposti i risultati ottenuti da varie simulazioni FEA eseguite mediante il software Abaqus CAE e lanciate attraverso un codice Python. Il codice è stato creato appositamente per definire le caratteristiche dell'RVE e delle simulazioni (descritte nel Capitolo 3), ovvero la geometria della beam con difetti interni e superficiali, le proprietà del materiale AlSi10Mg e le condizioni al contorno per le analisi agli elementi finiti. In questo modo, mediante cicli for annidati, è possibile studiare un parametro alla volta, variandone il valore e analizzando i risultati ottenuti in termini di curva tensione-deformazione della beam.

La geometria di base utilizzata per l'RVE è quella descritta nel Paragrafo 3.1 (cilindro con diametro pari a 1.4 mm e altezza di 2 mm), mentre verranno fatte variare le caratteristiche relative alla difettosità interna e superficiale. I parametri che risulteranno influenti sulle proprietà meccaniche della beam verranno utilizzati per definire un piano sperimentale di analisi FE (*Design of experiments* D.O.E.) e, di conseguenza, per ottenere un database da usare per le simulazioni della cella lattice intera su LS-Dyna e per l'addestramento della rete neurale.

4.1 Dimensione difetto

Il primo parametro preso in considerazione è la dimensione del difetto, il quale è stato valutato analizzando ventuno varianti di beam con differenti D_{max} per il difetto interno. Sono stati utilizzati i diametri relativi alle 21 classi di difetto della cella 2x2 (già descritti in Tabella 3.5), ovvero la tipologia di struttura lattice che verrà utilizzata per le simulazioni su LS-Dyna.

Le analisi sono state effettuate a parità delle altre condizioni di difettosità, descritte in Tabella 4.1.

Posizione difetto (x, y, z)	Orientamento difetto	Aspect ratio difetto	Periodo rugosità	Ampiezza rugosità
0, 0, 0	Orizzontale	1	200 µm	27.75 µm

Tabella 4. 1 - Caratteristiche di difettosità costanti per le analisi FE al variare della dimensione del difetto interno

In Figura 4.1 sono presenti tre esempi fra le casistiche analizzate, rappresentati dal difetto più piccolo (62 µm), uno di dimensione intermedia (309 µm) e quello più grande (631 µm).

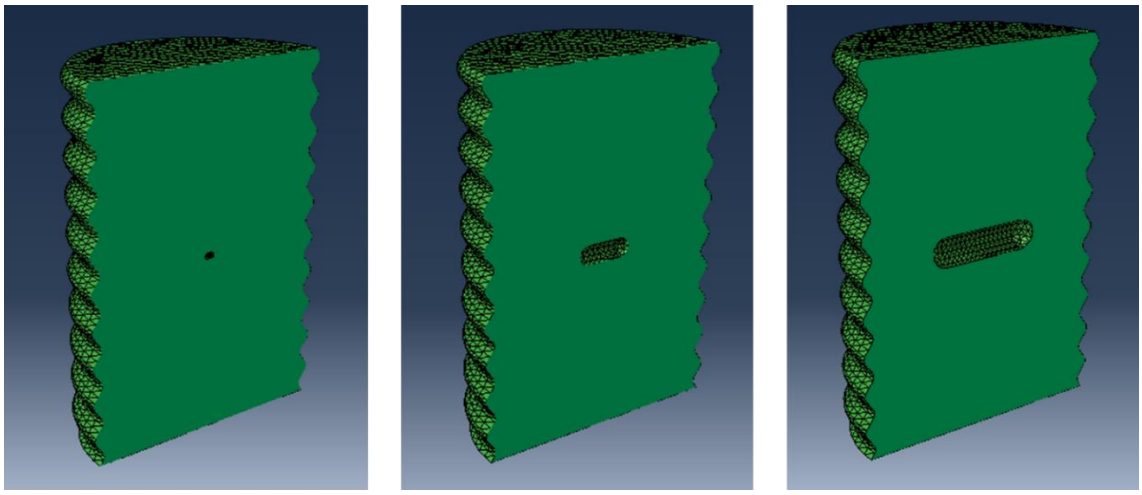

Figura 4. 1 - Tre diversi dimensioni di difetto interno, da sinistra a destra D_{max} pari a 62 µm, 309 µm e 631 µm

Dopo aver eseguito le simulazioni FEA su Abaqus, con le condizioni già descritte nel Paragrafo 3.2, sono state estratte le 21 curve omogeneizzate relative ai differenti valori di D_{max} della porosità interna, ottenendo il fascio di andamenti presente in Figura 4.2.

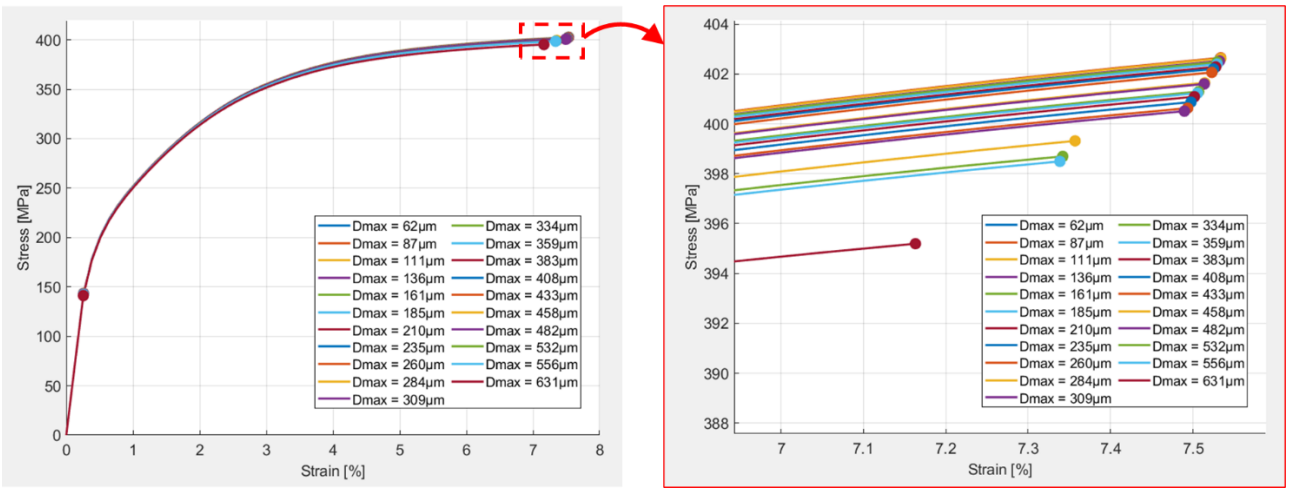

Figura 4. 2 - Curve omogeneizzate per diverse dimensioni di difetto nell'RVE, con zoom sulla zona di rottura

In Tabella 4.2 sono visibili i valori di modulo di Young (E), stress di rottura (*Ultimate Tensile Stress* UTS) e deformazione massima (ε_{max}) per i tre esempi presenti in Figura 4.1, con evidenziate le variazioni percentuali di tali proprietà rispetto al caso di difetto più piccolo.

D_max	62 µm	309 µm	Var. % fra 62 µm e 309 µm	631 µm	Var. % fra 62 µm e 631 µm
E [GPa]	55,370	55,314	-0,101	55,101	-0,486
UTS [MPa]	402,63	401,60	-0,255	395,19	-1,846
ε_max [%]	7,53	7,51	-0,256	7,16	-4,910

Tabella 4. 2 - Proprietà curva omogeneizzata dell'RVE all'aumentare delle dimensioni del difetto

Si può evincere come, all'aumentare della dimensione del difetto, si ottenga un peggioramento di tali caratteristiche della beam, soprattutto andando verso difetti più grandi, rendendo la dimensione del difetto uno dei parametri più influenti sul comportamento meccanico dell'RVE.

Il motivo principale di questo decremento delle proprietà meccaniche è da ricercare nella riduzione della sezione di materiale resistente per la beam. Infatti, aumentare il diametro del difetto equivale ad aumentare la sua superficie su un piano ortogonale alla direzione del carico applicato all'RVE (spostamento di trazione pari a 0.2 mm in direzione Y), come si può notare in Figura 4.3.

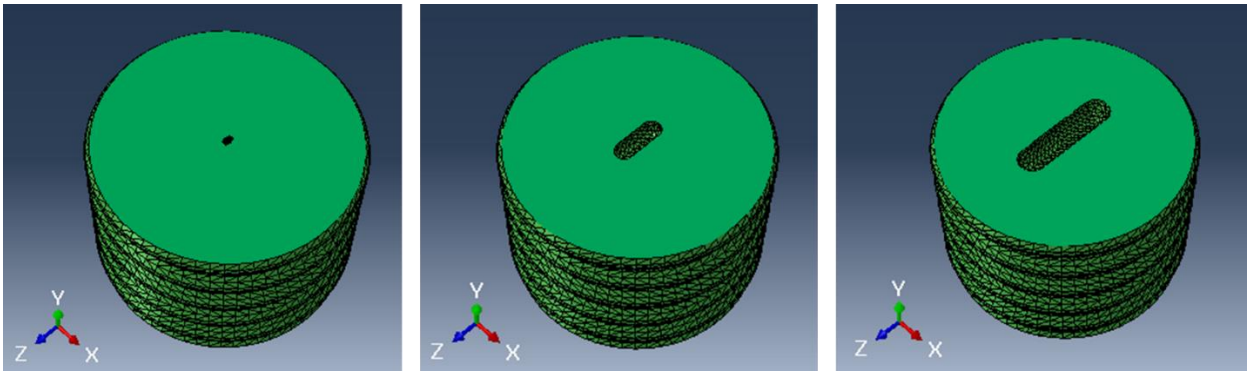

Figura 4. 3 - Diminuzione della sezione resistente dell'RVE all'aumentare del diametro del difetto

4.2 Posizione difetto

Il secondo parametro analizzato è la posizione della porosità interna lungo il piano XZ, tralasciando la traslazione lungo Y, poiché un difetto vicino alla base del cilindro (microscala) sarebbe comunque in posizione intermedia nella struttura lattice (macroscala), in quanto formata da tanti elementi beam collegati fra loro.

Come per il caso precedente, le analisi sono state effettuate a parità delle altre condizioni di difettosità, descritte in Tabella 4.3. Per questo confronto (e anche per i successivi), è stato scelto arbitrariamente il difetto di dimensione massima ($D_{max} = 631$ µm).

Dimensione difetto (D_{max})	Orientamento difetto	Aspect ratio difetto	Periodo rugosità	Ampiezza rugosità
631 µm	Orizzontale	1	200 µm	27.75 µm

Tabella 4. 3 - Caratteristiche di difettosità costanti per le analisi FE al variare della posizione del difetto interno

In Figura 4.4 sono presenti le tre casistiche analizzate, una col difetto in posizione centrale (X, Y, Z = 0, 0, 0), una in cui è posizionato vicino alla superficie della beam in direzione X e una analogamente in posizione Z; per quest'ultime sono stati considerati 70 µm fra la fine del cilindro e la faccia del difetto ortogonale alla direzione considerata. Come si può evincere dalle immagini estratte da Abaqus (sempre in Figura 4.4), la forma ellissoidale fa sì che ci siano entrambe le estremità del difetto vicino alla superficie dell'RVE nel caso di traslazione lungo X, mentre solo di un vertice per il caso lungo Z.

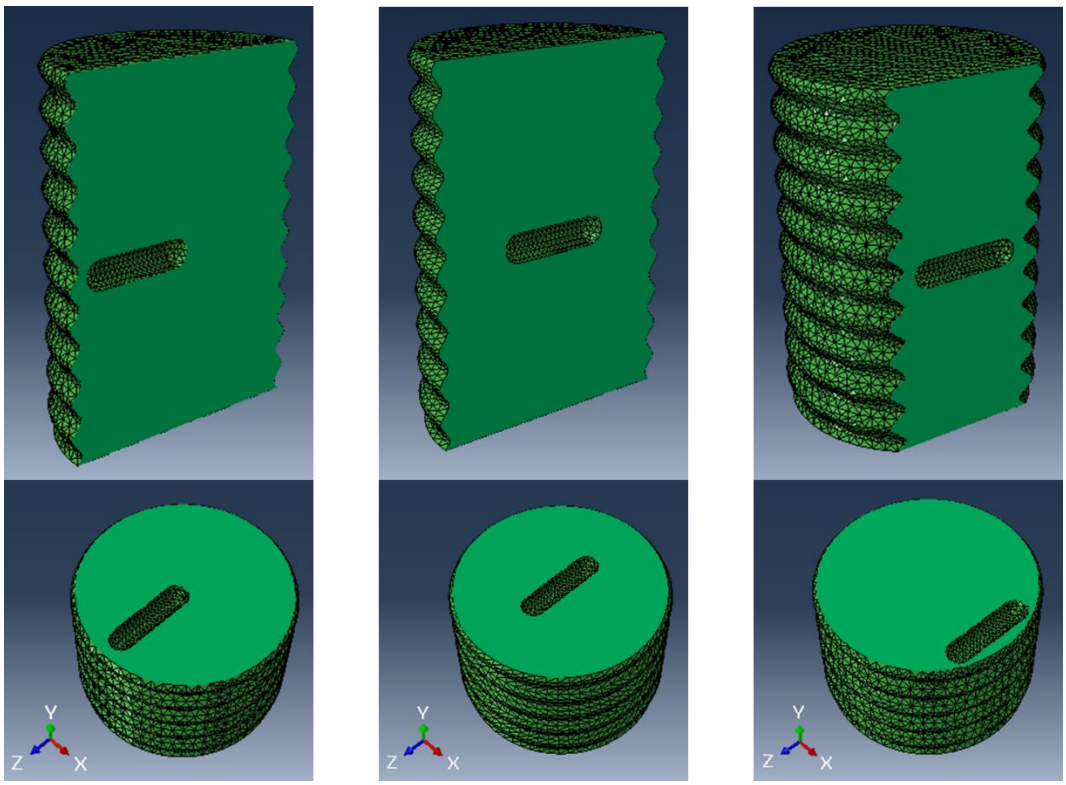

Figura 4. 4 - Tre posizioni considerate per il difetto interno: traslato lungo Z (sinistra), centrato (centro) e traslato lungo X (destra)

In Figura 4.5 sono visibili le curve omogeneizzate, ottenute in seguito alle analisi agli elementi finiti per le tre casistiche: si vede subito come ci sia una certa somiglianza fra i tre andamenti, anche visualizzando la zona di rottura con circa lo stesso livello di zoom utilizzato per il caso di diversa dimensione del difetto.

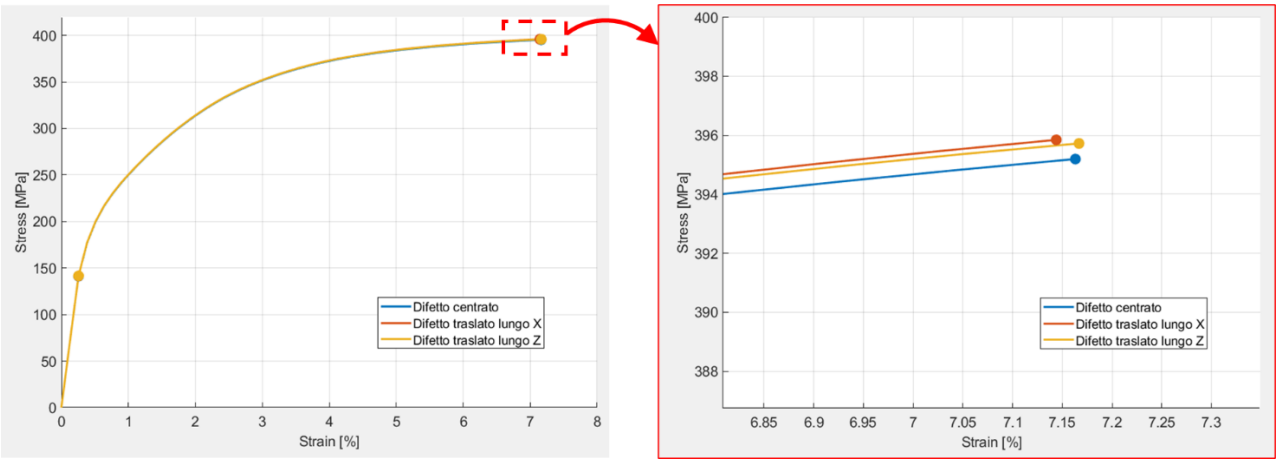

Figura 4. 5 - Curve omogeneizzate per diverse posizioni del difetto nell'RVE, con zoom sulla zona di rottura

Come dall'andamento delle curve, anche dai valori in Tabella 4.4 si evince come, fra i tre casi, non ci sia una differenza significativa tale da poter essere presa in considerazione in questo lavoro di tesi. Infatti, questo confronto conferma quanto detto in precedenza: l'aspetto più importante per queste analisi è la sezione resistente dell'RVE, la quale non varia se si modifica unicamente la posizione del difetto lungo il piano XZ; in questo modo il comportamento meccanico dell'RVE non viene influenzato significativamente.

Posizione difetto	Centrato	Traslato lungo X	Var. % fra centrato e lungo X	Traslato lungo Z	Var. % fra centrato e lungo Z
E [GPa]	55,101	55,190	0,162	55,134	0,060
UTS [MPa]	395,19	395,85	0,165	395,72	0,134
ε_max [%]	7,16	7,14	-0,268	7,17	0,052

Tabella 4. 4 - Proprietà curva omogeneizzata dell'RVE variando la posizione del difetto

Inoltre, si è notato come queste minime differenze siano visibili solo per la classe di difetto maggiore, mentre per dimensioni (anche di poco) minori gli andamenti risultano essere nettamente più simili fra loro, rendendo trascurabile l'influenza della posizione del difetto per il comportamento meccanico delle beam.

Tale parametro, quindi, non verrà preso in considerazione per la creazione del database per l'addestramento della rete neurale e per l'assegnazione delle proprietà microscopiche al modello macroscala.

4.3 Orientamento difetto

Un ulteriore parametro studiato è l'orientamento del difetto ellissoidale all'interno dell'RVE, a parità delle altre condizioni di difettosità, descritte in Tabella 4.5.

Dimensione difetto (D_{max})	Posizione difetto (x, y, z)	Aspect ratio difetto	Periodo rugosità	Ampiezza rugosità
631 µm	0, 0, 0	1	200 µm	27.75 µm

Tabella 4. 5 - Caratteristiche di difettosità costanti per le analisi FE al variare dell'orientamento del difetto interno

I tre casi analizzati (Figura 4.6) sono dati dal difetto orientato in maniera orizzontale (come le porosità considerate finora), obliqua (con asse principale ruotato di 45° rispetto a quello del cilindro) e verticale (asse parallelo a quello della beam).

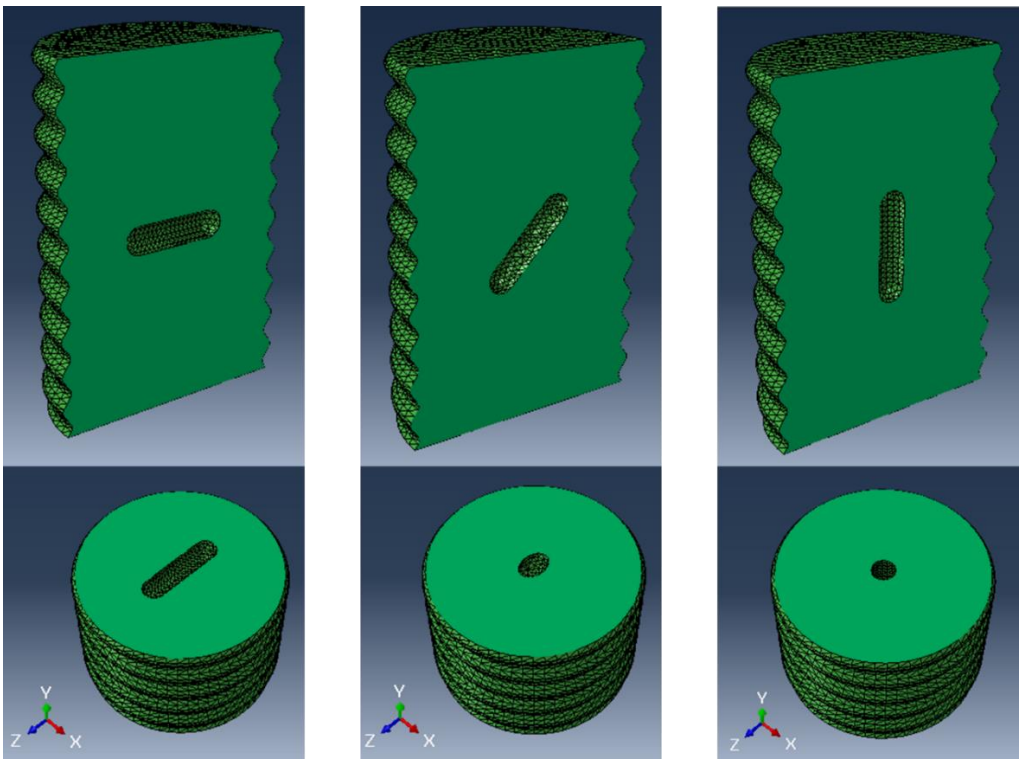

Figura 4. 6 - Tre orientamenti considerati per il difetto interno: orizzontale (sinistra), obliquo (centro) e verticale (destra)

Le curve omogeneizzate estratte in seguito alle analisi su Abaqus sono presenti in Figura 4.7. Dallo zoom nella zona di rottura e dai valori delle proprietà meccaniche principali, presenti in Tabella 4.6, si può evincere come il caso di difetto orizzontale sia quello più critico, mentre l'orientamento verticale rappresenti il caso con caratteristiche migliori e il difetto posto in maniera obliqua fornisce un comportamento intermedio.

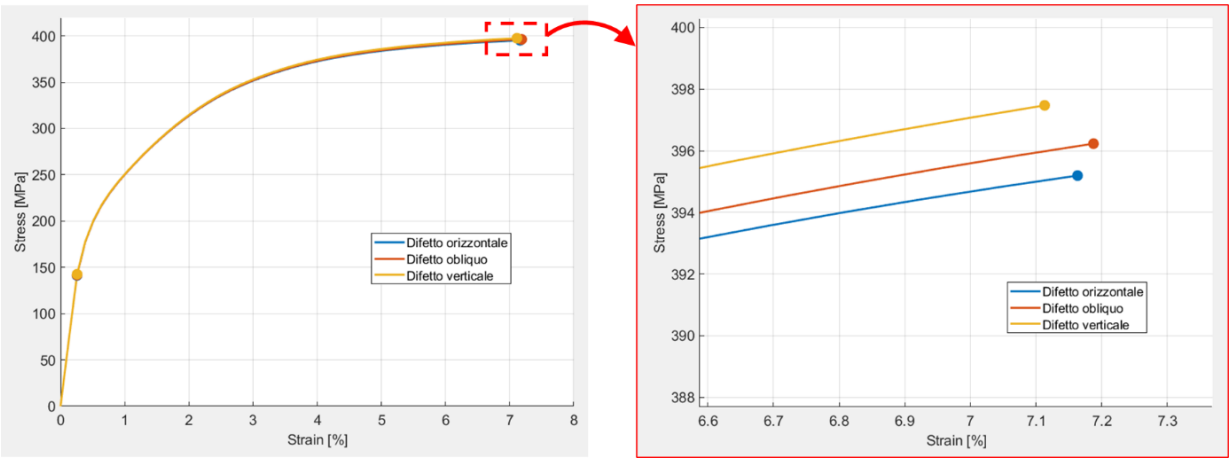

Figura 4. 7 - Curve omogeneizzate per diversi orientamenti del difetto nell'RVE, con zoom sulla zona di rottura

Orientamento difetto	Orizzontale	Obliquo	Var. % fra orizzontale e obliquo	Verticale	Var. % fra orizzontale e verticale
E [GPa]	55,101	55,163	0,113	55,265	0,298
UTS [MPa]	395,19	396,23	0,262	397,47	0,576
ε_max [%]	7,16	7,19	0,341	7,11	-0,699

Tabella 4. 6 - Proprietà curva omogeneizzata dell'RVE variando l'orientamento del difetto

Anche per queste analisi, la differenza è dovuta principalmente alla variazione di sezione resistente per la beam, come evidenziato dalla vista in sezione presente nella precedente Figura 4.6. Tuttavia, è da notare come per il caso con difetto verticale ci sia un decremento della deformazione massima, probabilmente dovuto al fatto che l'asse principale del difetto è nella stessa direzione della prova di trazione, rendendo più critico l'allungamento della beam durante le analisi numeriche.

Dunque, da tale studio, l'orientamento del difetto risulta essere un parametro influente sul comportamento omogeneizzato dell'RVE e perciò verrà preso in considerazione per la formazione del piano sperimentale definitivo.

4.4 Aspect ratio

L'ultimo parametro riguardante il difetto interno è l'aspect ratio (già descritto nel Paragrafo 3.4.2), il quale non viene estrapolato direttamente dalle scansioni Micro-CT, ma viene ricavato dal rapporto fra i PCA (Principal Component Analysis) estratti. L'Analisi dei Componenti Principali è una tecnica statistica utilizzata per identificare le direzioni (componenti) principali lungo le quali i dati variano di più. Le lunghezze delle componenti principali rappresentano le direzioni di massima varianza nei dati tridimensionali del difetto e possono essere usate per calcolare l'aspect ratio del difetto. In particolare, il rapporto fra la seconda e la terza componente principale (PC2 e PC3) rappresenta l'aspect ratio fra i due semi-assi minori del difetto ellissoidale.

È stata, quindi, calcolata la media fra tutti i rapporti PC2/PC3 per ogni difetto trovato mediante la scansione Micro-CT. Questa operazione ha fornito un valore medio di AR pari a circa 1.5; dunque, al fine di studiare l'influenza di tale parametro, è stato fatto un confronto fra tre valori di aspect ratio: 1, 1.5 e 2.

Le prime analisi FE, per confrontare tali valori di AR, sono state eseguite a parità delle caratteristiche presenti in Tabella 4.7, coinvolgendo le tipologie di beam presenti in Figura 4.8.

Dimensione difetto (D_{max})	**Posizione difetto (x, y, z)**	**Orientamento difetto**	**Periodo rugosità**	**Ampiezza rugosità**
631 µm	0, 0, 0	Orizzontale	200 µm	27.75 µm

Tabella 4. 7 - Caratteristiche di difettosità costanti per il primo studio sulla variazione di aspect ratio

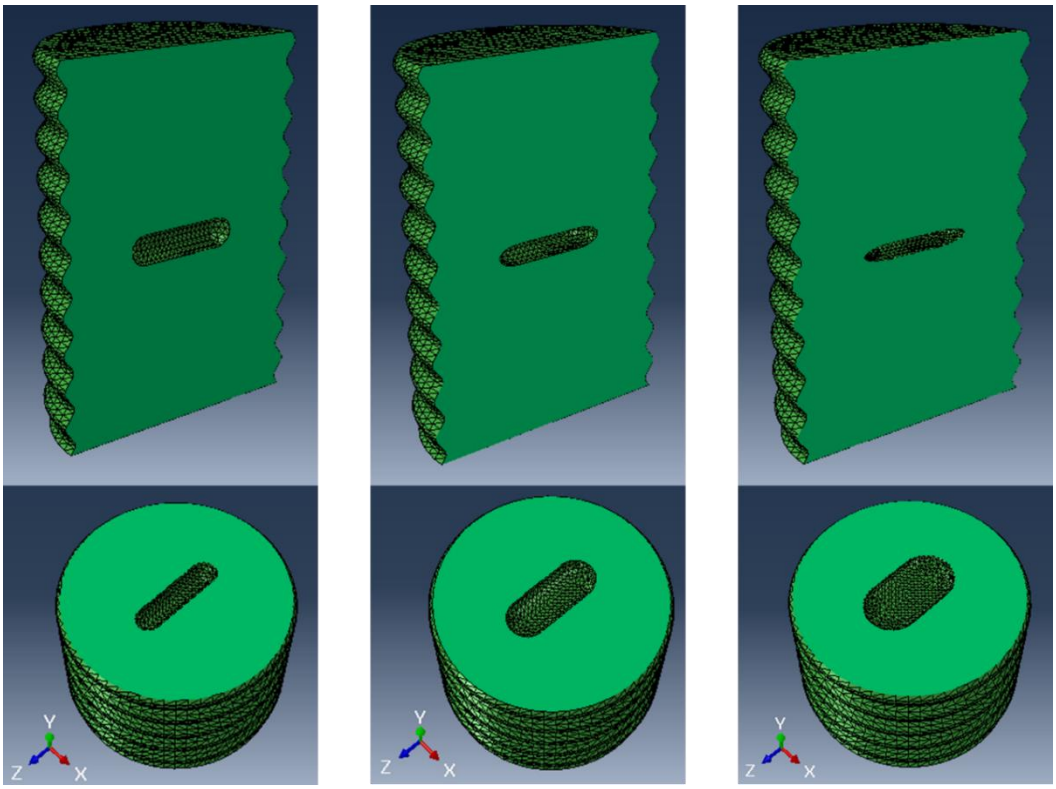

Figura 4. 8 - Tre casi considerati per difetto interno orizzontale da 631 µm: AR = 1 (sinistra), AR = 1.5 (centro) e AR = 2 (destra)

Dallo zoom sulla zona di rottura delle curve omogeneizzate (Figura 4.9) e dalle proprietà in Tabella 4.8 è evidente come l'incremento del valore di aspect ratio determini una decisa diminuzione sia dell'UTS che dell' ε_{max}.

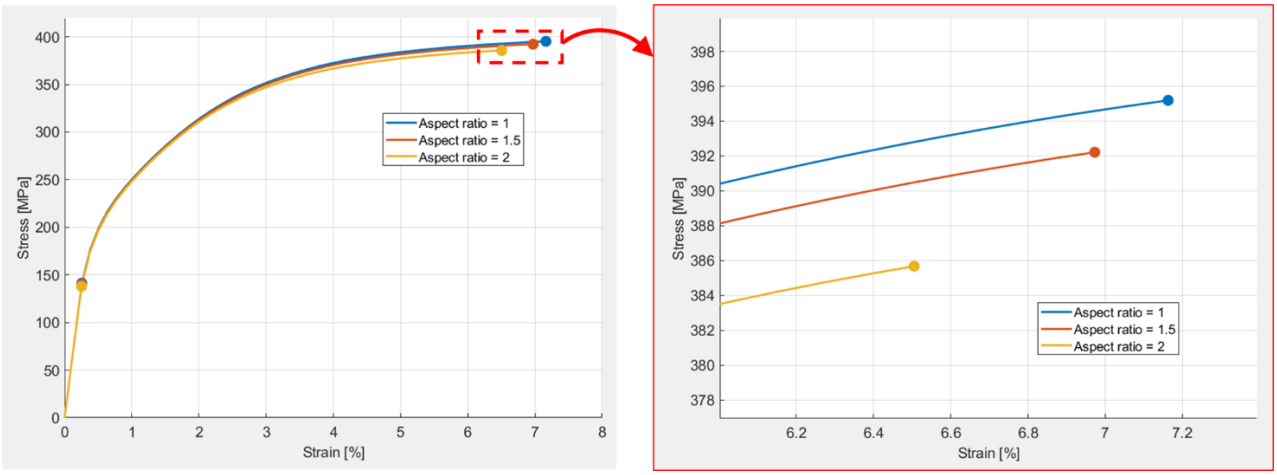

Figura 4. 9 - Curve omogeneizzate per il primo studio sulla variazione di aspect ratio, con zoom sulla zona di rottura

Difetto orizzontale 631 µm	AR = 1	AR = 1.5	Var. % fra AR = 1 e AR = 1.5	AR = 2	Var. % fra AR = 1 e AR = 2
E [GPa]	55,101	54,960	-0,256	54,792	-0,560
UTS [MPa]	395,19	392,22	-0,753	385,68	-2,408
ε_max [%]	7,16	6,97	-2,655	6,51	-9,183

Tabella 4. 8 - Proprietà curva omogeneizzata dell'RVE per il primo studio sulla variazione di aspect ratio

Questo comportamento è dovuto al significativo decremento della sezione resistente nella zona del difetto, in quanto si ha un "appiattimento" del difetto sul piano ortogonale alla direzione della prova di trazione.

Confermata quindi, in generale, l'influenza dell'aspect ratio sul comportamento dell'RVE, sono state eseguite delle analisi più dettagliate su tale parametro, al fine di capire se la sua influenza fosse rilevante anche per diverse tipologie di orientamento e dimensione del difetto.

4.4.1 Aspect ratio al variare dell'orientamento del difetto

Sono state eseguite le simulazioni FE variando sia l'aspect ratio che l'orientamento del difetto, ottenendo, per il caso di difetto obliquo, risultati analoghi a quelli estratti dal caso con difetto orizzontale. Per il difetto verticale, invece, sono stati ottenuti dei risultati differenti in seguito alle analisi sugli RVE raffigurati in Figura 4.10 (mantenendo quindi costanti le caratteristiche di difettosità presenti in Tabella 4.9).

Dimensione difetto (D_{max})	**Posizione difetto (x, y, z)**	**Orientamento difetto**	**Periodo rugosità**	**Ampiezza rugosità**
631 µm	0, 0, 0	Verticale	200 µm	27.75 µm

Tabella 4. 9 - Caratteristiche di difettosità costanti per lo studio sulla variazione dell'aspect ratio su difetti verticali

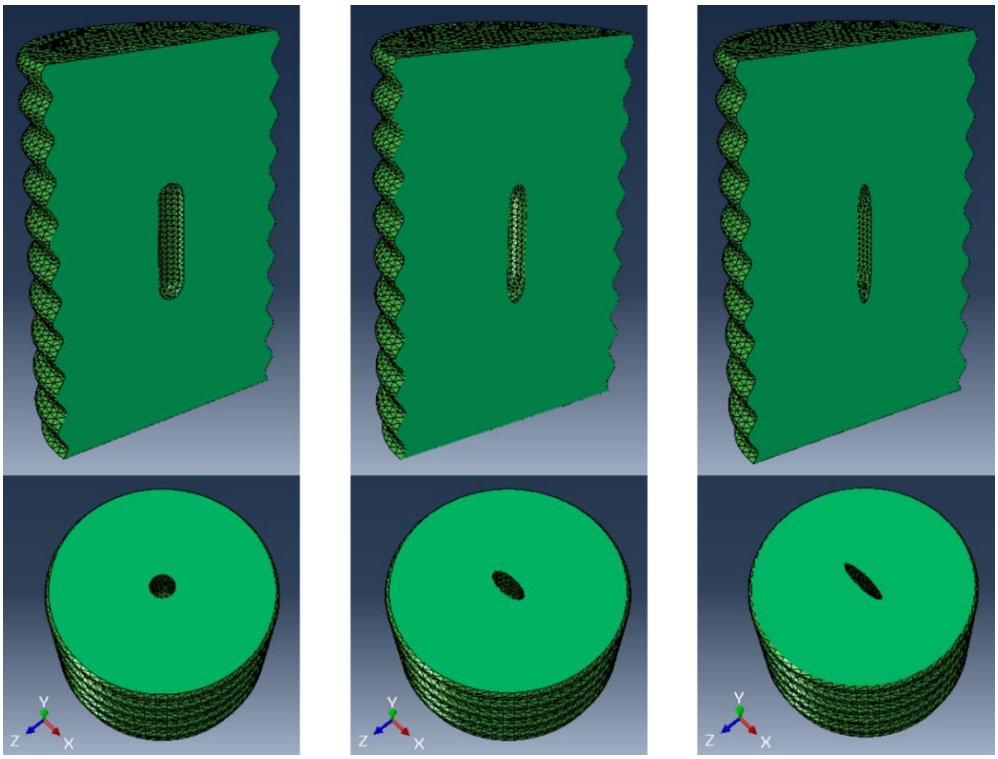

Figura 4. 10 - Tre casi considerati per difetto interno verticale da 631 µm: AR = 1 (sinistra), AR = 1.5 (centro) e AR = 2 (destra)

Dalla Figura 4.11 e dalla Tabella 4.9 è evidente come la variazione di aspect ratio non incida significativamente sul difetto verticale. Tale comportamento è giustificato dal fatto che, modificando il valore di AR, si abbia una certa similitudine fra l'aumento del semiasse del difetto lungo X (da Figura 4.10) e la diminuzione lungo Z, mantenendo praticamente costante la sezione resistente all'altezza della porosità interna.

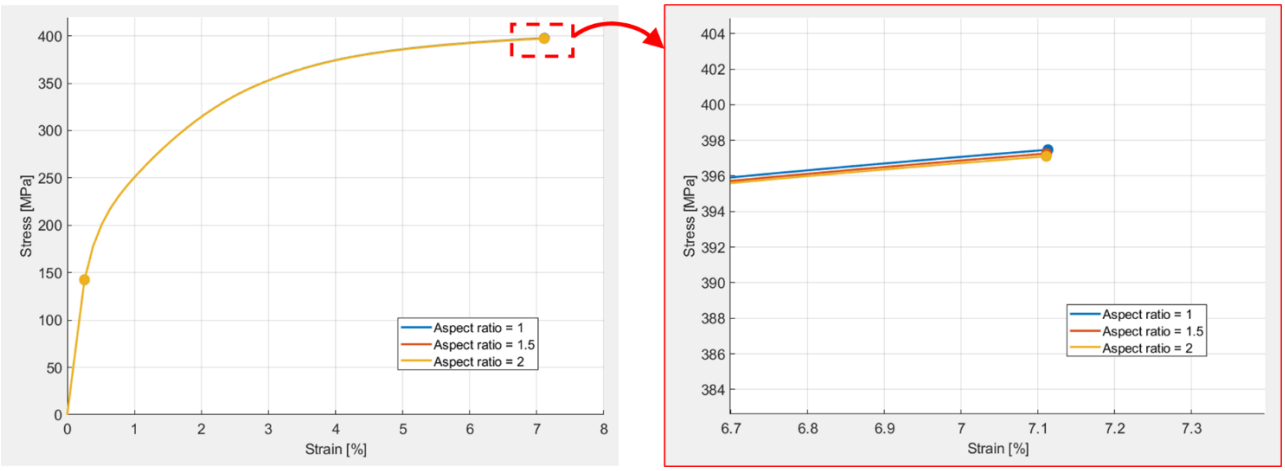

Figura 4. 11 - Curve omogeneizzate per lo studio sulla variazione dell'aspect ratio su difetti verticali, con zoom sulla zona di rottura

Difetto verticale 631 µm	AR = 1	AR = 1.5	Var. % fra AR = 1 e AR = 1.5	AR = 2	Var. % fra AR = 1 e AR = 2
E [GPa]	55,265	55,260	-0,009	55,260	-0,009
UTS [MPa]	397,47	397,26	-0,054	397,10	-0,093
ε_max [%]	7,11	7,11	-0,027	7,11	-0,031

Tabella 4. 10 - Proprietà curva omogeneizzata dell'RVE per lo studio sulla variazione dell'aspect ratio su difetti verticali

A causa di tale comportamento, durante la creazione del piano sperimentale saranno considerati per i casi con orientazione verticale solo quelli con AR = 1, poiché, da quanto appena esposto, risultano essere già una buona approssimazione per i difetti interni con medesimo orientamento e con AR = 1.5 e AR = 2.

4.4.2 Aspect ratio al variare della dimensione del difetto

È stata condotta un'ulteriore analisi per poter capire se l'aspect ratio fosse significativamente influente su tutte le classi di difetto, o solo da una certa dimensione in poi. Sono state quindi effettuate diverse analisi partendo dal valore di D_{max} più piccolo, ed è stato riscontrato come dalla prima all'ottava classe di difetto non ci sia una sostanziale differenza utilizzando un valore diverso di AR. I primi risultati degni di nota sono stati trovati per la classe n°9 (D_{max} = 260 µm), quindi variando i valori di AR e mantenendo costanti i parametri in Tabella 4.11.

Dimensione difetto (D_{max})	**Posizione difetto (x, y, z)**	**Orientamento difetto**	**Periodo rugosità**	**Ampiezza rugosità**
260 µm	0, 0, 0	Orizzontale	200 µm	27.75 µm

Tabella 4. 11 - Caratteristiche di difettosità costanti per lo studio sulla variazione dell'aspect ratio per dimensioni minori del difetto interno

Come si può evincere dalle curve omogeneizzate (Figura 4.12) e dalle proprietà ottenute (Tabella 4.12), le differenze fra i tre casi non sono ancora eccessive, ma comunque visibili e, soprattutto, con un andamento che inizia a rispecchiare il comportamento realistico che sarà più evidente per difetti più grandi, ovvero un peggioramento delle caratteristiche meccaniche all'aumentare del valore di aspect ratio (a differenza delle analisi effettuate fino alla classe n°8).

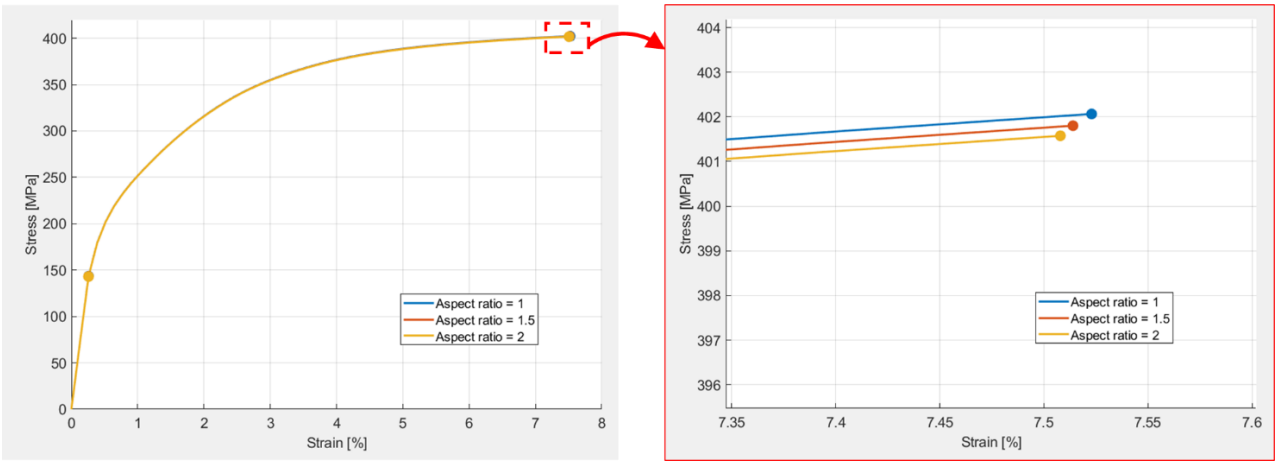

Figura 4. 12 - Curve omogeneizzate per lo studio sulla variazione di aspect ratio per dimensioni minori del difetto interno, con zoom sulla zona di rottura

Difetto orizzontale 260 µm	AR = 1	AR = 1.5	Var. % fra AR = 1 e AR = 1.5	AR = 2	Var. % fra AR = 1 e AR = 2
E [GPa]	55,342	55,320	-0,040	55,296	-0,083
UTS [MPa]	402,07	401,80	-0,065	401,58	-0,122
ε_max [%]	7,523	7,514	-0,120	7,508	-0,199

Tabella 4. 12 - Proprietà curva omogeneizzata dell'RVE per lo studio sulla variazione di aspect ratio per dimensioni minori del difetto interno

Le analisi effettuate portano quindi a trascurare l'influenza dell'aspect ratio dalla prima all'ottava classe di difetto; tali dimensioni verranno quindi inserite all'interno del piano sperimentale solo mediante il caso con AR = 1 e variando gli altri parametri influenti sul comportamento dell'RVE.

4.5 Rugosità superficiale

Come già anticipato nel Paragrafo 3.3, la rugosità viene simulata mediante un profilo sinusoidale lungo la beam, il quale avrà una certa ampiezza e un certo periodo. È stato già mostrato come una sinusoide con periodo pari a 200 µm e ampiezza di 27.75 µm riesca ad approssimare abbastanza bene il comportamento di una beam con una superficie reale estratta mediante scansioni Micro-CT. Nelle pagine successive, quindi, verrà analizzato come la modifica di tali parametri influenzi il comportamento omogeneizzato dell'RVE.

4.5.1 Periodo rugosità sinusoidale

Si è partiti dalla variazione del periodo per il profilo sinusoidale, mantenendo costanti gli altri parametri presenti in Tabella 4.13.

Dimensione difetto (D_{max})	Posizione difetto (x, y, z)	Orientamento difetto	Aspect ratio difetto	Ampiezza rugosità
631 µm	0, 0, 0	Orizzontale	1	27.75 µm

Tabella 4. 13 - Caratteristiche di difettosità costanti per le analisi FE al variare del periodo della rugosità simulata

Non avendo dati sperimentali su cui basarsi per questo parametro, si è scelto di confrontare il periodo del profilo sinusoidale utilizzato finora (200 µm) con due altri casi in cui è stato prima ridotto fino a 125 µm e poi aumentato fino a 250 µm. Fra le varie possibilità, sono stati confrontati questi valori, in quanto tutti e tre hanno permesso di ottenere una sinusoide completa considerando la lunghezza dell'RVE pari a 2 mm. Infatti, partendo dalla tipologia con periodo più piccolo a quella con periodo più grande, si ottengono rispettivamente 16, 10 e 8 cicli di sinusoide ripetuti in maniera identica uno dopo l'altro, come visibile in Figura 4.13.

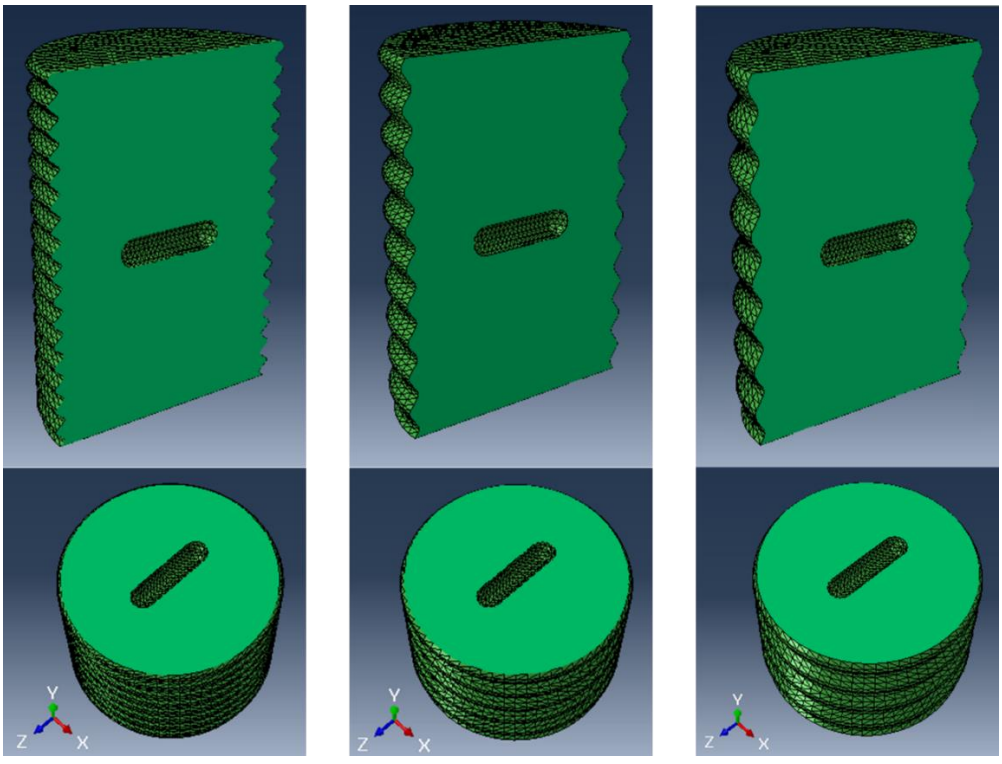

Figura 4. 13 - Tre periodi di rugosità considerati: 125 µm (sinistra), 200 µm (centro) e 250 µm (destra)

Osservando le curve omogeneizzate in Figura 4.14 e le proprietà della beam in Tabella 4.14 si nota come all'aumentare del periodo della sinusoide si registri un deciso incremento per la deformazione massima, nonostante un valore circa costante o una leggera decrescita per l'UTS.

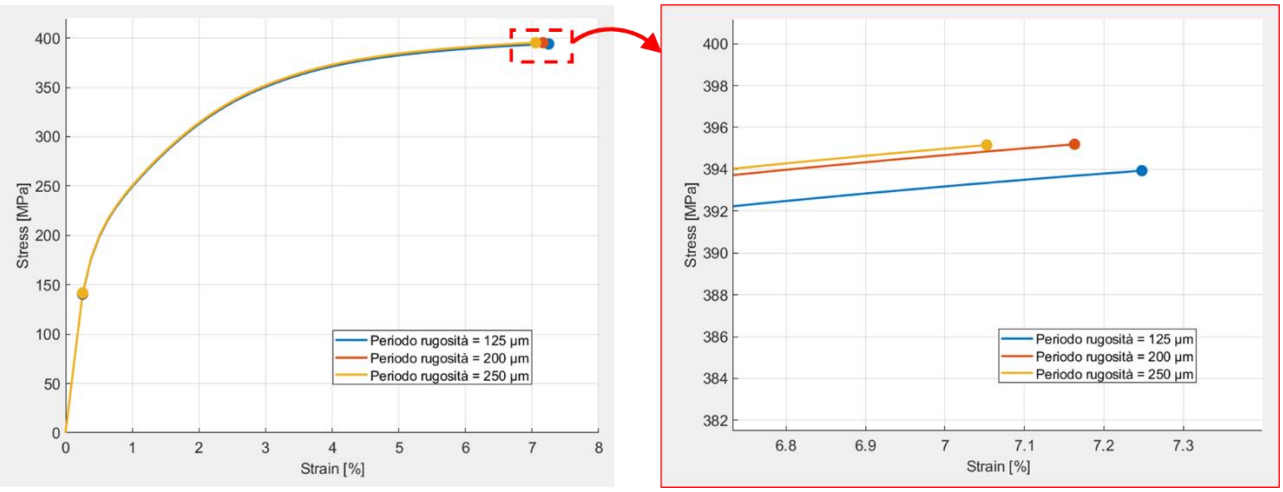

Figura 4. 14 - Curve omogeneizzate per diversi periodi di profilo sinusoidale, con zoom sulla zona di rottura

Periodo rugosità	200 µm	125 µm	Var. % fra 200 µm e 125 µm	250 µm	Var. % fra 200 µm e 250 µm
E [GPa]	55,101	55,077	-0,043	55,104	0,006
UTS [MPa]	395,19	393,93	-0,319	395,16	-0,009
ε_max [%]	7,16	7,25	1,182	7,05	-1,541

Tabella 4. 14 - Proprietà curva omogeneizzata dell'RVE variando il periodo del profilo sinusoidale

Poiché la sezione resistente nell'intorno del difetto si può considerare circa costante per queste tre tipologie di RVE, tale caratteristica non sembra essere la causa delle proprietà appena descritte. Quindi, questo comportamento può essere giustificato dal fatto che, durante le analisi numeriche, una superficie con un profilo sinusoidale più accentuato possa favorire la deformazione della struttura, analogamente, per grandi linee, al comportamento di una molla in trazione.

Tali andamenti giustificano l'inserimento di questa proprietà fra quelle influenti per la creazione del database da utilizzare per le simulazioni macroscala e per l'addestramento dell'algoritmo di Machine Learning.

4.5.2 Ampiezza rugosità sinusoidale

Infine, sono state eseguite analisi FE variando l'ampiezza della sinusoide che approssima il profilo della beam, lasciando costanti gli altri parametri di difettosità in Tabella 4.15.

Dimensione difetto (D_{max})	Posizione difetto (x, y, z)	Orientamento difetto	Aspect ratio difetto	Periodo rugosità
631 µm	0, 0, 0	Orizzontale	1	200 µm

Tabella 4. 15 - Caratteristiche di difettosità costanti per le analisi FE al variare dell'ampiezza della rugosità simulata

In Figura 4.15 sono raffigurati i tre casi studiati, uno dei quali è corrispondente all'ampiezza di 27.75 µm usata fino ad ora e già descritta nel Paragrafo 3.3 per il modello di RVE di base. Per la rugosità delle altre due beam sono state usate un'ampiezza di 11.6 µm, corrispondente alla deviazione media aritmetica (Ra) estratta dalle prove sperimentali e presente in Tabella 3.3, e un'altra di 55.4 µm, pari all'altezza massima del profilo (Rz) e anch'essa in Tabella 3.3.

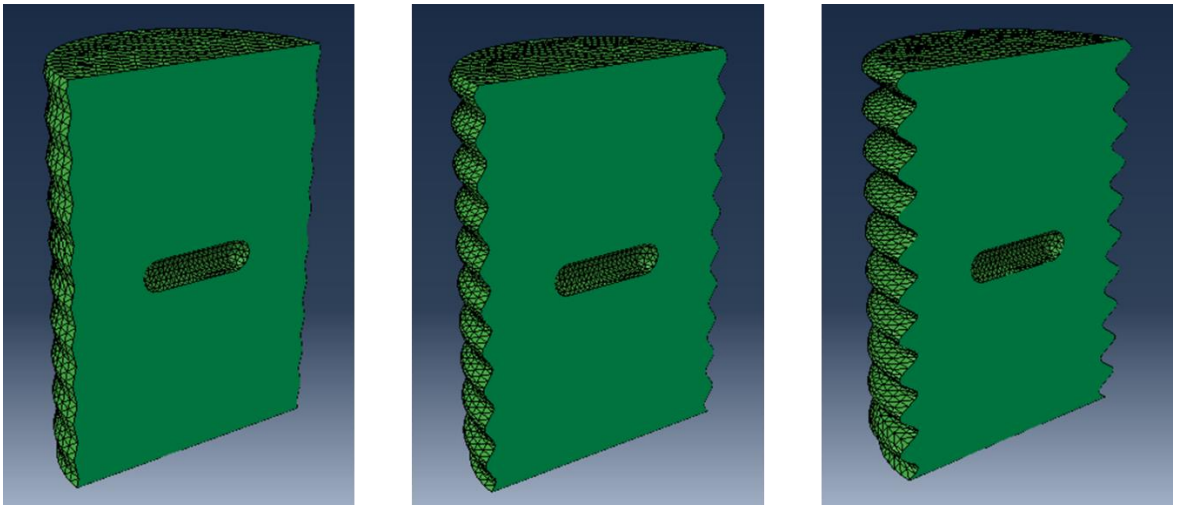

Figura 4. 15 - Tre ampiezze di rugosità considerate: 11.6 µm (sinistra), 27.75 µm (centro) e 55.4 µm (destra)

Le curve omogeneizzate estratte da queste tre analisi (Figura 4.16) e le conseguenti proprietà degli RVE (Tabella 4.16) evidenziano come, in generale, le caratteristiche dell'RVE peggiorino all'aumentare dell'ampiezza di rugosità.

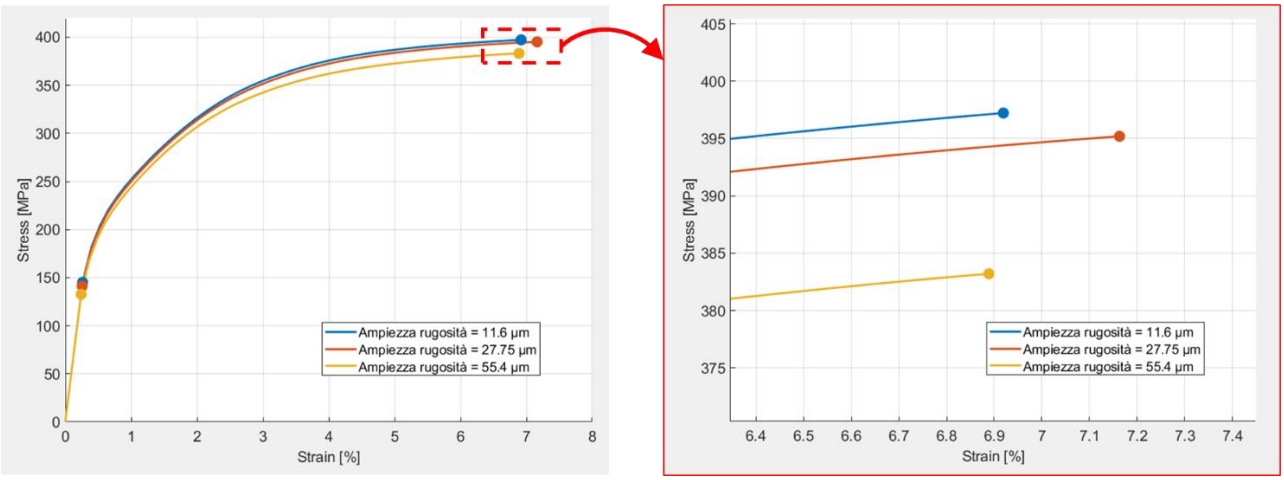

Figura 4. 16 - Curve omogeneizzate per diverse ampiezze di profilo sinusoidale, con zoom sulla zona di rottura

Ampiezza rugosità	27.75 µm	11.6 µm	Var. % fra 27.75 µm e 11.6 µm	55.4 µm	Var. % fra 27.75 µm e 55.4 µm
E [GPa]	55,101	55,234	0,242	55,103	0,004
UTS [MPa]	395,19	397,22	0,514	383,21	-3,032
ε_max [%]	7,16	6,92	-3,404	6,89	-3,828

Tabella 4. 16 - Proprietà curva omogeneizzata dell'RVE variando l'ampiezza del profilo sinusoidale

Come nei casi precedenti, il primo motivo dell'abbassamento della curva omogeneizzata è dovuto alla diminuzione di sezione resistente all'altezza del difetto, anche se minimo, visibile dalla sezione dell'RVE in Figura 4.17 (area resistente evidenziata in rosso per cogliere meglio la leggera variazione nei tre casi). Inoltre, all'aumentare dell'ampiezza della sinusoide si ha anche una maggiore intensità delle tensioni per gli elementi della mesh in corrispondenza dei picchi del profilo sinusoidale. Si nota, tuttavia, un particolare incremento del ε_{max} per il caso con ampiezza pari a 27.75 µm, la cui giustificazione può essere trovata nell'aver trovato un buon compromesso fra tale parametro e il periodo corrente (200 µm) per ottenere, dalle analisi numeriche, un buon risultato in termini di deformazione.

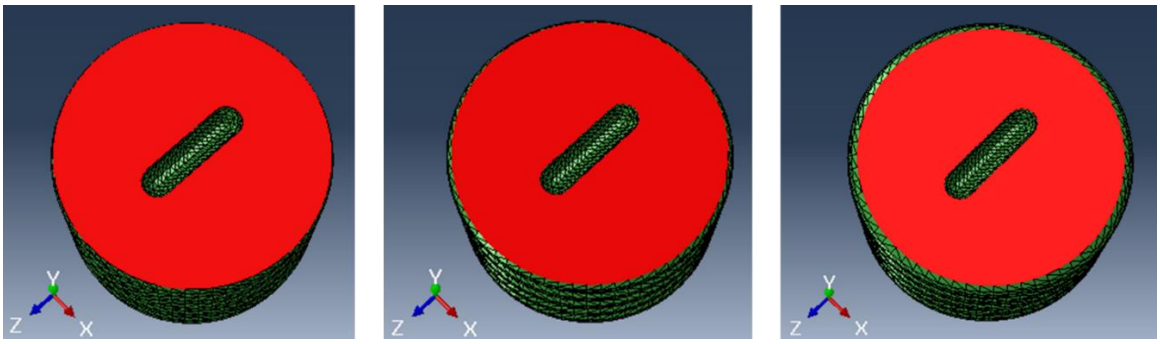

Figura 4. 17 – Leggera diminuzione della sezione resistente dell'RVE all'aumentare dell'ampiezza del profilo sinusoidale

Quindi, a causa del doppio effetto appena descritto che porta ad una significativa influenza di questi tre valori (realistici in quanto derivanti da prove sperimentali) per la rugosità superficiale della beam, essi verranno utilizzati per ampliare il database definitivo.

4.6 Piano sperimentale (D.O.E.)

Una volta che sono stati identificati i parametri che hanno un'influenza significativa sulla risposta omogeneizzata della beam, si è passato alla definizione di un piano sperimentale (detto anche *Design of experiments*). Questo consiste nell'eseguire una serie di simulazioni su Abaqus CAE, al fine di ottenere differenti curve omogeneizzate sulla base di tutte le possibili combinazioni dei parametri di difettosità risultati influenti (dagli studi descritti nei paragrafi precedenti).

Per eseguire tali simulazioni sono stati utilizzati tre differenti codici Python, uno per ogni aspect ratio considerato, essendo l'unico parametro per cui non sono state simulate tutte le combinazioni con ciascuna delle sue tre varianti. In Tabella 4.17 sono schematizzate tutte le varianti considerate, e si può notare come siano stati trascurate le prime otto classi di difetto e l'orientazione verticale (90°) per AR = 1.5 e AR = 2, a causa di quanto ottenuto dagli studi nei paragrafi precedenti.

AR	Classi difetto	Orientamento difetto	Ampiezza sinusoide	Periodo sinusoide
1	da 1 a 21 = **21 var.**	0° , 45° , 90° = **3 var.**	125 µm , 200 µm , 250 µm = **3 var.**	11.6 µm , 27.75 µm , 55.4 µm = **3 var.**
1,5	da 9 a 21 = **13 var.**	0° , 45° = **2 var.**	125 µm , 200 µm , 250 µm = **3 var.**	11.6 µm , 27.75 µm , 55.4 µm = **3 var.**
2	da 9 a 21 = **13 var.**	0° , 45° = **2 var.**	125 µm , 200 µm , 250 µm = **3 var.**	11.6 µm , 27.75 µm , 55.4 µm = **3 var.**

Tabella 4. 17 - Varianti considerate per la definizione del piano sperimentale

Lanciando i tre codici Python per gli altrettanti valori di aspect ratio, vengono eseguite in totale 1035 simulazioni mediante Abaqus CAE, pari al numero di tutte le possibili combinazioni date dal piano sperimentale, come visibile in Tabella 4.18 (ripresa da Excel).

Aspect ratio		1	1,5	2
N° varianti	Classi difetto	21	13	13
	Orientamento difetto	3	2	2
	Ampiezza rugosità	3	3	3
	Periodo rugosità	3	3	3
N° combinazioni per ogni AR		567	234	234
N° COMBINAZIONI TOTALI		**1035**		

Tabella 4. 18 - Numero di tutte le combinazioni date dal piano sperimentale

In ciascun codice sono presenti quattro funzioni per definire le diverse fasi della simulazione: la modellazione della beam con i suoi difetti interni e superficiali, l'assegnazione delle proprietà del materiale AlSi10Mg, l'applicazione delle condizioni al contorno e l'esecuzione della simulazione FEA. Dopo la scrittura di tali funzioni, nello script è presente un ciclo for annidato che considera tutte le possibili combinazioni date dall'unione di ciascun valore di diametro massimo, orientamento del difetto, ampiezza e periodo del profilo sinusoidale dell'RVE (mentre l'AR, come già detto, viene variato usando tre codici diversi). Per ognuna delle 1035 combinazioni vengono richiamate le funzioni definite all'inizio dello script, eseguendo tutte le simulazioni una dopo l'altra fino all'esaurimento delle varianti. Inoltre, per ogni simulazione, viene calcolata ed estratta la curva omogeneizzata, per poi essere salvata in una cartella contenente tutti gli andamenti ottenuti. Infine, prima di passare all'analisi successiva, vengono eliminati i file di simulazione '*.odb*', '*.prt*' e '*.sim*', così da evitare, durante le analisi, problemi dovuti all'eccessivo spazio occupato sul disco. In Appendice è presente un estratto del codice per AR = 1, preso come esempio.

Con i risultati ottenuti è stato possibile creare un database formato da un elevatissimo numero di curve omogeneizzate, che verrà utilizzato prima per le analisi macroscala (sulla cella lattice intera) mediante LS-Dyna, e poi per l'addestramento del modello di Machine Learning per la previsione della risposta a compressione della struttura lattice.

5. Analisi macroscala su cella lattice

Come visto nel capitolo precedente, le analisi effettuate in scala locale su RVE con difettosità hanno permesso di ottenere curve sforzo-deformazione dipendenti da vari parametri, legati alla porosità interna e alla rugosità superficiale delle beam. Ora, dunque, è possibile passare alla scala globale per associare queste proprietà alla cella lattice intera.

La struttura reticolare è composta da travi interconnesse fra loro, le quali presentano una sezione trasversale circolare con un diametro di 1.4 mm. Il campione analizzato in questo studio ha una struttura cubica con dimensioni di 20 x 20 x 20 mm (cella 2x2), analogo a quello su cui sono state effettuate le scansioni Micro-CT (presentate nel Paragrafo 3.4).

Utilizzando il software LS-Dyna (specializzato in analisi agli elementi finiti non lineari), la struttura lattice è stata discretizzata con una mesh di elementi beam 1D. Ogni elemento, che rappresenta una parte di una trave, è definito mediante una CARD con vari parametri, come ad esempio le proprietà dell'RVE e gli ID dei nodi che connettono ciascun elemento.

Quindi, mediante questo approccio multiscala, la difettosità della singola trave viene valutata su scala locale analizzando le beam, per poi associare alla struttura lattice globale la risposta ottenuta. In Figura 5.1 è visibile la cella 2x2 oggetto di studio, formata da diversi elementi 1D con differente CARD.

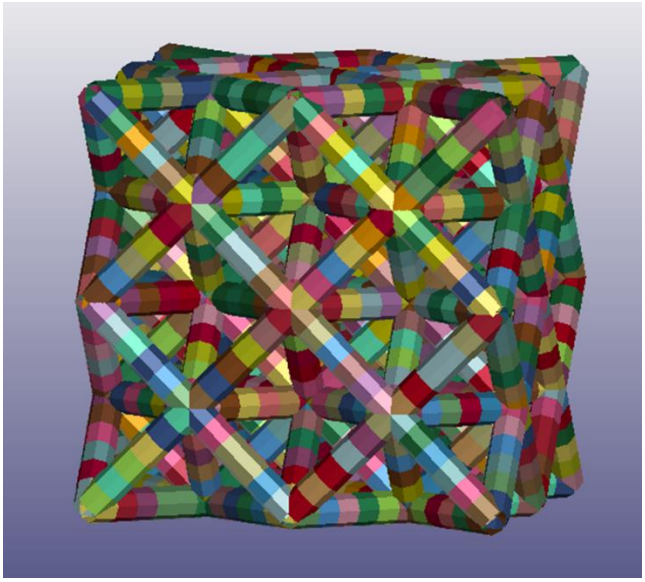

Figura 5. 1 - Cella 2x2 generata mediante LS-Dyna con elementi 1D con CARD differenti

5.1 Creazione database

Per la modellazione della cella lattice su LS-Dyna è necessario associare a ciascun elemento che la compone una curva tensione-deformazione fra quelle ottenute dalle simulazioni eseguite durante il piano sperimentale. Tali caratteristiche meccaniche dovranno essere pescate in maniera casuale, ma assegnando loro una certa probabilità di essere scelte in base alle frequenze ottenute (per ciascuna classe di difetto) dalle scansioni micro-CT della cella 2x2, presenti in Tabella 3.5.

Inoltre, considerando quanto descritto nel Paragrafo 4.6, per ciascuna classe di difetto esistono 81 combinazioni disponibili, considerati i parametri di aspect ratio, orientamento del difetto interno, ampiezza e periodo della rugosità superficiale (studiati nel capitolo precedente). Avendo tre varianti per ciascuno di questi quattro parametri, si avranno $3^4 = 81$ combinazioni, che moltiplicate per le 21 classi di difetto forniscono 1701 possibili curve da poter assegnare agli elementi 1D che compongono la struttura.

Tuttavia, per il D.O.E. sono state eseguite 1035 combinazioni, in quanto alcune varianti erano risultate superflue da eseguire poiché aventi risultati molto simili ad altre già presenti nel piano sperimentale. Per la creazione del database, però, è stato necessario inserire le curve mancanti, effettuando una copia degli andamenti in similitudine. In particolare, sulla base dei risultati ottenuti nel Capitolo 4 e da quanto riportato in Tabelle 4.17 e 4.18, sono state eseguite due principali tipologie di copie:

- per le prime 8 classi di difetto: a parità di dimensione, orientamento del difetto, ampiezza e periodo di rugosità superficiale per le curve relative ad AR=1.5 e AR=2 sono state prese le copie delle rispettive curve con AR=1;
- per i difetti verticali: a parità di dimensione del difetto, ampiezza e periodo di rugosità superficiale, tutte le curve con orientamento verticale per AR=1.5 e AR=2 vengono ottenute mediante la copia dei rispettivi andamenti per AR=1.

In totale quindi, alle 1035 curve del piano sperimentale, verranno aggiunte 666 curve di copia, date da: 288 copie dovute soltanto alle similitudini fra le prime 8 classi di difetto (non considerando i difetti verticali), 234 copie dovute unicamente agli andamenti simili per difetti verticali (per le restanti 13 classi di difetto) e, infine, 144 copie dovute ad entrambi i casi.

In questo modo si ottiene il database completo con 1701 curve, che viene utilizzato per assegnare le diverse caratteristiche meccaniche agli elementi 1D costituenti la cella lattice per la simulazione FEA su LS-Dyna.

È stato tracciato un plot raffigurante tutte le curve del database ottenuto, presente in Figura 5.2. Da tale grafico si può evincere la grande variabilità di caratteristiche meccaniche che potranno essere attribuite agli elementi del modello macroscala a causa delle diverse tipologie di difettosità, nonostante sia stato assegnato lo stesso materiale AlSi10Mg a ciascuno di loro.

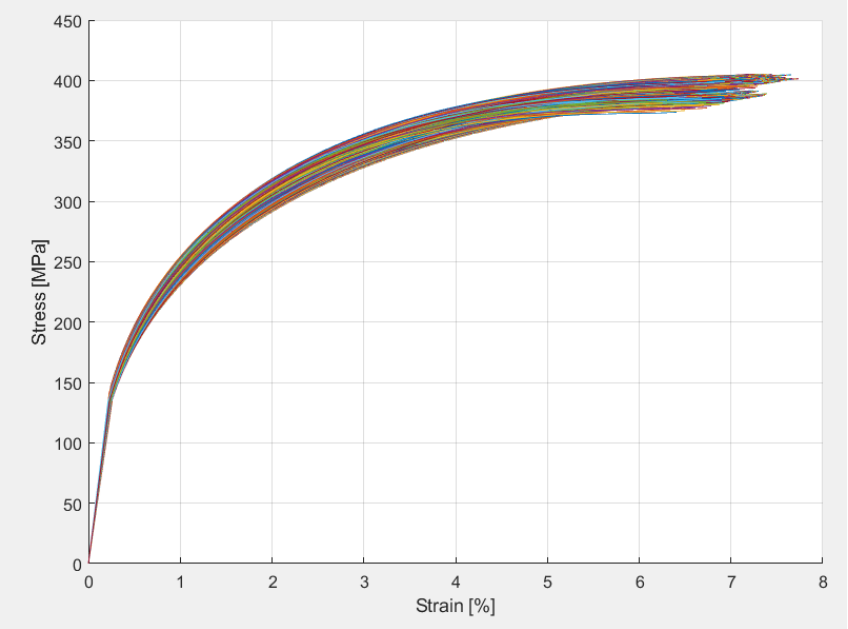

Figura 5. 2 - Curve presenti nel database intero

5.2 Modello struttura lattice

Le curve omogeneizzate, formate da coppie di colonne stress-strain, vengono raccolte tutte in una tabella in formato *.mat*, che viene poi letta da un codice MATLAB che genera diversi file *.k* (da *keyword,* usati per gli input in LS-Dyna) contenenti informazioni su materiali, parti ed elementi che riguardano il modello per le simulazioni FEA. Nelle prossime pagine è quindi spiegata la struttura e le funzionalità di tale codice MATLAB.

Inizialmente si carica nel workspace un vettore ‘frequency’, contenente tutti i valori di probabilità per ciascuna classe di difetto (già visti in Tabella 3.5), che viene poi utilizzato nella seguente linea di codice:

```
pid = randsample(2:(n_classi+1), totale_elementi, true, frequency)
```

in cui sono presenti anche il numero di classi di difetto (21 come già visto) e il numero di elementi 1D che compongono il modello di cella lattice, pari a 2304 in questo caso di studio. Questa linea di codice utilizza la funzione `randsample` per generare un vettore ‘pid’, con lunghezza pari al numero di elementi 1D presenti nella struttura e formato da valori campionati casualmente fra le possibili classi di difetto, dove le probabilità di campionamento sono determinate dal vettore ‘frequency’.

Dopo aver fatto ciò, viene eseguita la parte di codice in Figura 5.3, utile per assegnare un unico ID a ciascuna curva nel database.

```
pid_unique = unique(pid);
pid_def = zeros(1, totale_elementi);
for i = 1:length(pid_unique)
    aspect_ratio = randsample(1:n_aspect_ratio, length(pid(pid == pid_unique(i))), true);
    orientation = randsample(1:n_angles, length(pid(pid == pid_unique(i))), true);
    amp_rug = randsample(1:n_amp_rug, length(pid(pid == pid_unique(i))), true);
    period_rug = randsample(1:n_period_rug, length(pid(pid == pid_unique(i))), true);
    pid_def(1, pid == pid_unique(i)) = ...
        (pid_unique(i) - 2) * n_aspect_ratio * n_angles  * n_amp_rug * n_period_rug + ...
        (aspect_ratio - 1) * n_angles * n_amp_rug * n_period_rug + ...
        (orientation - 1)  * n_amp_rug * n_period_rug + ...
        (amp_rug - 1) * n_period_rug + ...
        period_rug + 3 ; % +3 perchè devo far partire il conto dei pid da 4
end
```

Figura 5. 3 - Da MATLAB, ciclo for per l'assegnazione casuale di una curva dal database a ciascun elemento del modello lattice

Inizialmente si usa il comando `unique` per creare un array nominato ‘pid_unique’ contenente i valori unici presenti in ‘pid’. Tale comando, infatti, viene utilizzato in MATLAB per identificare gli elementi unici in un array o una matrice, restituendo un vettore con solo questi elementi distinti.

Poi si passa al campionamento di valori casuali (con il comando `randsample`) per aspect ratio, orientamento, ampiezza e periodo della rugosità superficiale, basato sul numero di volte che il valore di '`pid_unique(i)`' appare nel vettore '`pid`'.

Dunque, mediante un ciclo for, viene generato un vettore nominato '`pid_def`', al cui interno viene associato ogni valore unico presente nell'array '`pid`', cioè '`pid_unique(i)`', ad un indice calcolato sulla base del campionamento casuale dei parametri specificati (`aspect_ratio`, `orientation`, `amp_rug`, `period_rug`). In particolare, viene assegnato un unico ID da 4 a 1705 ad ognuna delle 1701 combinazioni, ordinate allo stesso modo sia per la tabella di input e sia per gli indici, ovvero prima in ordine di classe di difetto, e poi di aspect ratio, orientamento, ampiezza e periodo di rugosità superficiale.

Questi indici servono, quindi, a contraddistinguere la tipologia di curva che è stata assegnata a ciascun elemento mediante il vettore '`pid_def`', e tale numerazione parte dall'ID 4 poiché gli indici precedenti servono a definire i due muri rigidi usati durante la simulazione FEA per la prova di compressione sul modello lattice.

Per i file keyword che verranno generati, i 1701 differenti ID saranno associati a due parametri fondamentali, denominati "*mid*" e "*pid*": questi rappresentano rispettivamente l'ID assegnato a uno specifico materiale (e quindi ad una determinata curva del database) e l'ID assegnato alla parte (beam) corrispondente a quel materiale.

Dopo aver fatto ciò, viene creato il primo dei file da eseguire per le analisi agli elementi finiti, ovvero "*element_beam.k*". Al suo interno vengono inseriti uno alla volta i valori presenti nel nuovo vettore '`pid_def`' per ciascun elemento del modello lattice per LS-Dyna, ognuno dei quali identificato da un indice *"eid"* (ID elemento). Inoltre, per ogni riga sono presenti anche gli indici *n1* e *n2*, che definiscono i nodi, e quindi le estremità di ogni elemento mediante le quali le beam sono interconnesse fra loro.

In Figura 5.4 è presente un estratto del file, in cui si può osservare l'associazione di ciascun elemento ad un valore di "*pid*" che varia casualmente da 4 a 1705, seguendo i comandi dati dal ciclo for in Figura 5.3 descritto in precedenza.

```
*ELEMENT_BEAM
$#   eid     pid      n1      n2      n3     rt1     rr1     rt2     rr2   local
     163     343     352     353       0       0       0       0       0       2
     164     521     353     355       0       0       0       0       0       2
     165     102     355     357       0       0       0       0       0       2
     166     514     357     359       0       0       0       0       0       2
     167     296     359     361       0       0       0       0       0       2
     168      85     361     194       0       0       0       0       0       2
    1687     125     194    3655       0       0       0       0       0       2
    1688     195    3655    3657       0       0       0       0       0       2
    1689     610    3657    3659       0       0       0       0       0       2
    1690     633    3659    3661       0       0       0       0       0       2
    1691     124    3661    3663       0       0       0       0       0       2
    1692     577    3663     701       0       0       0       0       0       2
    3229     582     194    6996       0       0       0       0       0       2
    3230     179    6996    6998       0       0       0       0       0       2
    3231     351    6998    7000       0       0       0       0       0       2
    3232     113    7000    7002       0       0       0       0       0       2
    3233     195    7002    7004       0       0       0       0       0       2
    3234     520    7004    1262       0       0       0       0       0       2
```

Figura 5. 4 - Estratto del file "element_beam.k" in cui ad ogni elemento del modello viene associato un valore di PID in maniera casuale

In seguito, si passa alla generazione del file "*material.k*", contenente diverse schede di materiale (material card) per ciascuna combinazione presente nel database.

In Figura 5.5, è presente un esempio di card del materiale (mid = 4, quindi corrispondente alla prima curva dal database), dove la variabile *mid* varia in modo incrementale, assumendo, come già descritto, un valore caratteristico per ognuna delle 1701 combinazioni. Un altro parametro importante è il *fail*, necessario per definire il cedimento per ciascuna scheda materiale e analogo al valore di deformazione ultima per la corrispondente curva omogeneizzata, ripresa dal database.

```
*MAT_PIECEWISE_LINEAR_PLASTICITY_TITLE
AlSi10Mg
$#     mid        ro         e        pr      sigy      etan      fail      tdel
         42.60000E-9   53000.0      0.33       0.0       0.0  0.073944       0.0
$#       c         p      lcss      lcsr        vp
       0.0       0.0         4         0       0.0
$$ HM Entries in Stress-Strain Curve =         8
$#    eps1      eps2      eps3      eps4      eps5      eps6      eps7      eps8
       0.0       0.0       0.0       0.0       0.0       0.0       0.0       0.0
$#     es1       es2       es3       es4       es5       es6       es7       es8
       0.0       0.0       0.0       0.0       0.0       0.0       0.0       0.0
```

Figura 5. 5 - Esempio di material card dal file "material.k"

Ad ognuna di queste card deve essere collegata la rispettiva curva stress-strain ripresa dal database; per fare ciò, il codice MATLAB prima legge il file di input contenente le curve, e poi riscrive nel file "*material.k*", nella sezione “DEFINE_CURVE” apposita, i valori di tensione e deformazione per la scheda corrispondente. Un esempio di tale sezione del file è presente in Figura 5.6, in cui si può vedere la parte iniziale della curva corrispondente a mid=4; si può notare come la colonna *a1* corrisponda ai valori di deformazione (asse X della curva), mentre la colonna *o1* (asse Y) rappresenti i corrispondenti valori di stress per la specificata tipologia di RVE.

```
*DEFINE_CURVE_TITLE
Plastic_Sig-Eps_Aluminum_3D
$#    lcid      sidr       sfa       sfo      offa      offo    dattyp     lcint
         4         0       1.0       1.0 -0.001327       0.0         0         0
$#              a1                  o1
          0.001327           73.727800
          0.002653          147.064000
          0.003975          182.794000
          0.005299          205.884000
          0.006621          222.845000
          0.007941          236.119000
          0.009260          247.641000
          0.010578          257.825000
          0.011894          267.462000
          0.013209          276.822000
          0.014522          285.603000
          0.015833          293.955000
          0.017143          301.928000
          0.018450          309.453000
          0.019756          316.507000
          0.021060          323.123000
          0.022363          329.326000
```

Figura 5. 6 - Esempio di curva omogeneizzata (non intera) riportata sul file "material.k"

Infine, il codice MATLAB genera un file "*part.k*" in cui le material card create sono unicamente associate ad una singola parte corrispondente.

In Figura 5.7, è raffigurata la scheda per la prima parte con un *pid* (ID parte) pari a 4, corrispondente alla prima tipologia di RVE. Un "*mid*" (ID materiale) unico pari a 4 è associato a pid=4. Le schede successive incrementano sia *pid* che *mid* contemporaneamente fino a raggiungere l'ultima combinazione presente nel database. Si può notare come ciò che rimane costante sia il *secid* (ID sezione): infatti, ne è presente solo uno per tutte le parti (e per tutti gli elementi) poiché il diametro della trave rimane costante a 1.4 mm.

```
*PART
$HWCOLOR COMPS       1       3
$#                                                                  title
MESH_Unit_Cell_04_3x3
$#     pid     secid       mid     eosid      hgid      grav    adpopt      tmid
         4         1         4         0         0         0         0         0
```

Figura 5. 7 - Material card associata a rispettiva parte su file "part.k"

Dopo aver generato con MATLAB i tre file *.k* appena descritti, si sono potute effettuare le analisi sul modello lattice in LS-Dyna simulando test di compressione quasi-statici. Tali prove vengono condotte applicando degli spostamenti a una parete rigida in contatto con il campione, la cui risposta a compressione viene valutata utilizzando come output le curve Forza-Spostamento estraibili dall'analisi agli elementi finiti.

Tali simulazioni vengono lanciate mediante un file keyword principale, in cui, mediante il comando INCLUDE, vengono richiamati i file "*element_beam.k*", "*material.k*" e "*part.k*", come si può vedere in Figura 5.8.

```
*INCLUDE
$#                                                              filename
element_beam.k
*INCLUDE
$#                                                              filename
material.k
*INCLUDE
$#                                                              filename
part.k
```

Figura 5. 8 - Comandi INCLUDE all'interno del file keyword principale

Sempre mediante lo stesso file *.k* principale, vengono create due pareti rigide parallele in contatto con la cella lattice (script in Figura 5.9): una parete inferiore, che rimane fissa per replicare il piano di supporto, e una parete superiore, a cui viene assegnata una legge di spostamento lineare per simulare numericamente la compressione del campione.

```
*BOUNDARY_PRESCRIBED_MOTION_RIGID
$#     pid       dof       vad      lcid        sf       vid     death     birth
         3         3         2         1      -1.0         01.00000E28       0.0
*BOUNDARY_SPC_SET
$#    nsid       cid      dofx      dofy      dofz     dofrx     dofry     dofrz
         1         0         1         1         1         1         1         1
*SET_NODE_LIST
$#     sid       da1       da2       da3       da4    solver
         1       0.0       0.0       0.0       0.0MECH
$#    nid1      nid2      nid3      nid4      nid5      nid6      nid7      nid8
     13153     13154     13155     13156     13157     13158     13159     13160
     13161     13162     13163     13164     13165     13166     13167     13168
     13169     13170     13171     13172     13173     13174     13175     13176
     13177     13178     13179     13180     13181     13182     13183     13184
     13185     13186     13187     13188         0         0         0         0
*BOUNDARY_SPC_SET
$#    nsid       cid      dofx      dofy      dofz     dofrx     dofry     dofrz
         2         0         1         1         0         1         1         1
*SET_NODE_LIST
$#     sid       da1       da2       da3       da4    solver
         2       0.0       0.0       0.0       0.0MECH
$#    nid1      nid2      nid3      nid4      nid5      nid6      nid7      nid8
     13189     13190     13191     13192     13193     13194     13195     13196
     13197     13198     13199     13200     13201     13202     13203     13204
     13205     13206     13207     13208     13209     13210     13211     13212
     13213     13214     13215     13216     13217     13218     13219     13220
     13221     13222     13223     13224         0         0         0         0
```

Figura 5. 9 – Definizione due pareti rigide per la simulazione FEA del test di compressione

Tale legge di spostamento è identificata da un indice *lcid* = 1 ed è determinata dall'estratto del file in Figura 5.10, in cui i due valori chiamati '*sfa*' e '*sfo*' scalano la curva di spostamento nel tempo, imponendo che dopo 0.0175 secondi il muro rigido si debba spostare di 17.5 mm. Quindi, la parete avrà una velocità di spostamento pari a 17.5 mm / 0.0175 s = 1000 mm/s, cioè di 1 m/s. Inoltre, la durata della simulazione viene impostata ad un valore di 3 ms, ottenendo (sulla base della legge di spostamento lineare del muro superiore) una compressione di 3 mm per la struttura lattice.

```
*DEFINE_CURVE_TITLE
Linear_Displacement
$#    lcid      sidr       sfa       sfo      offa      offo    dattyp     lcint
         1         0    0.0175      17.5       0.0       0.0         0         0
$#                a1                  o1
                0.0                 0.0
                1.0                 1.0
```

Figura 5. 10 – Definizione della legge di spostamento lineare per la parete rigida superiore

Importando su LS-Dyna il file keyword con tutte le informazioni appena descritte, si ottiene il modello visibile in Figura 5.11: esso è quindi composto dalla cella lattice formata dagli elementi 1D con diverse material card, su cui verranno simulati numericamente i test di compressione mediante la movimentazione della parete rigida superiore.

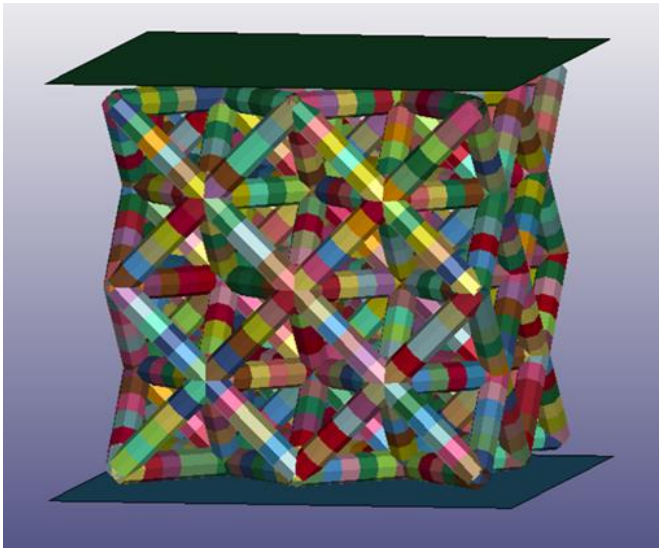

Figura 5. 11 - Modello della cella lattice in LS-Dyna per il test di compressione, con la presenza delle due pareti rigide

5.3 Risultati FEA e confronto con curve sperimentali

Dopo la generazione del modello, è ora possibile eseguire le analisi agli elementi finiti in LS-Dyna, seguite dal confronto fra la risposta a compressione numerica e quella sperimentale del componente.

In Figura 5.12 è presente la curva forza-spostamento per la cella 2x2, ottenuta sperimentalmente mediante prova di compressione quasi-statica: si può notare come al picco di forza avvenga la rottura di una prima beam appartenente alla struttura lattice, seguita da numerose piccole fratture di altre travi, le quali determinano l'andamento segmentato della curva.

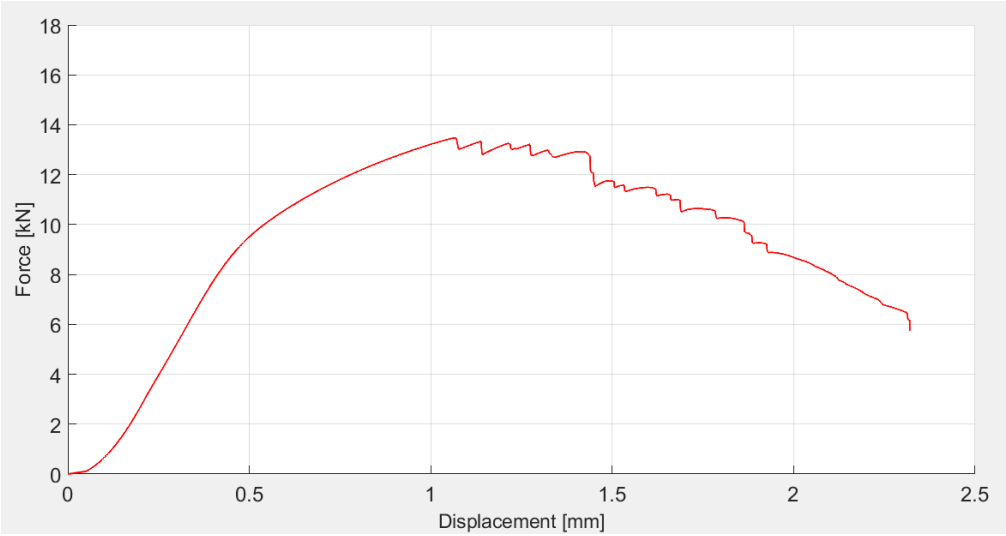

Figura 5. 12 - Curva sperimentale per prova di compressione quasi-statica su cella lattice 2x2

La prova di compressione quasi-statica è un test utilizzato per determinare le proprietà meccaniche di un materiale sottoponendolo a compressione lenta e controllata. Infatti, col termine "quasi-statica" si intende che la velocità di applicazione del carico è molto bassa, in modo che gli effetti inerziali siano trascurabili e il comportamento del materiale possa essere valutato in condizioni simili a quelle statiche.

La velocità di spostamento (della parete superiore) per tale test sperimentale si attesta attorno a 1 mm/min, quindi molto più lento della simulazione impostata su LS-Dyna mediante i file keyword: come detto in precedenza, le analisi FE vengono effettuate con una velocità di spostamento pari a 1 m/s, ottenendo una compressione della cella lattice di 3mm e una durata complessiva di 3ms per la prova.

Per ottenere lo stesso spostamento, ma con le velocità usate per il test sperimentale, sono necessari tre minuti di compressione, che per essere analizzati numericamente avrebbero bisogno di tempi eccessivi e ingestibili, mentre è possibile simulare le compressioni quasi-statiche mediante analisi FE a prescindere dalla velocità di compressione.

Infatti, come definito nel manuale del software LS-Dyna per una simulazione numerica quasi-statica, non è fondamentale che la velocità di simulazione del test sia uguale a quella sperimentale, ma è sufficiente che l'energia cinetica sia trascurabile rispetto all'energia interna, e quindi che gli effetti inerziali siano minimizzati.

Un approccio per ridurre il tempo di simulazione risiede quindi nell'applicare il carico più rapidamente rispetto al test sperimentale, ma assicurandosi che l'energia cinetica rimanga bassa. Ciò è possibile anche perché il modello di materiale utilizzato non considera effetti di strain rate nella risposta meccanica del materiale.

Eseguendo le simulazioni su Ls-Dyna vengono generati i file *d3plot*, in cui sono presenti, oltre ai risultati delle analisi, gli andamenti di energia cinetica ed energia interna durante le simulazioni. Prendendo una simulazione di esempio, sono stati tracciati i grafici relativi a tali andamenti (Figure 5.13 e 5.14), da cui si può notare come l'energia cinetica raggiunga al massimo dei picchi pari a 250 J, rendendola trascurabile rispetto a quella interna, per cui si ottengono anche valori di quasi 14 kJ, quindi superiori di due ordini di grandezza. Inoltre, si osserva un brusco aumento dell'energia cinetica in corrispondenza del cedimento delle beam nella struttura lattice (1.8 ms). Tale aumento è associato proprio al distacco di alcuni elementi trave. Fino al cedimento, l'energia cinetica è pari a 25 J, ovvero tre ordini di grandezza inferiori rispetto all'energia interna di deformazione della struttura lattice.

Sulla base di quanto detto precedentemente, questi risultati confermano che le simulazioni su LS-Dyna possono essere definite "quasi-statiche", così come i test sperimentali con cui saranno confrontate.

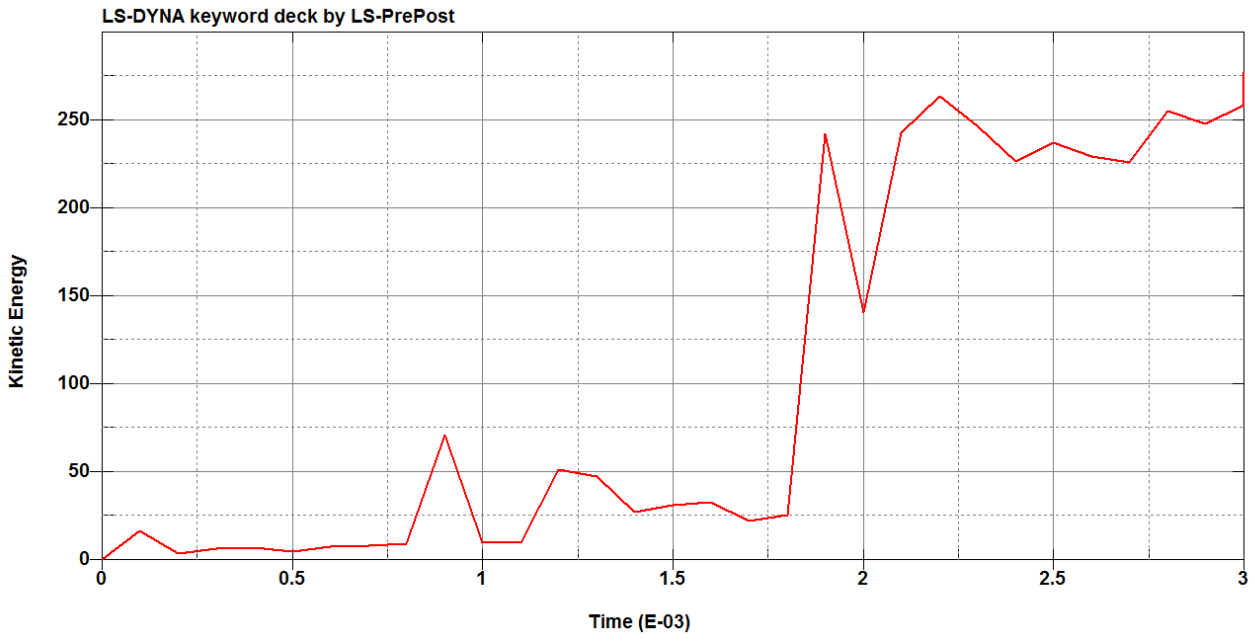

Figura 5. 13 – Andamento dell'energia cinetica durante una delle simulazioni effettuate su LS-Dyna

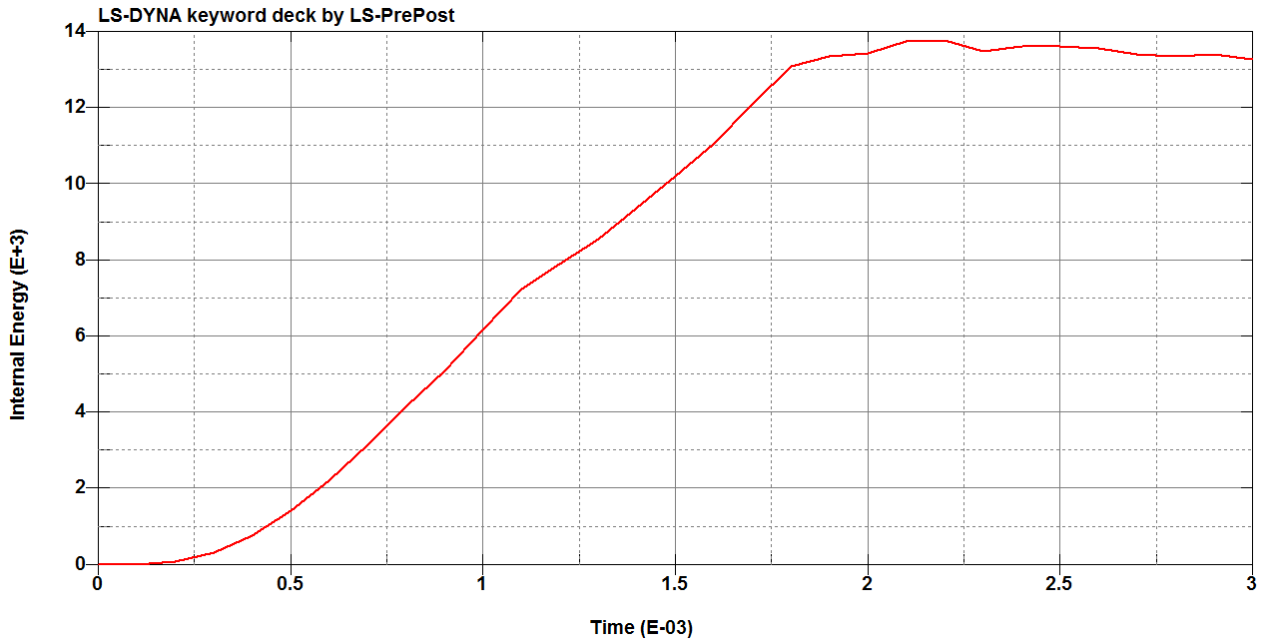

Figura 5. 14 – Andamento dell'energia interna durante una delle simulazioni effettuate su LS-Dyna

Aprendo i file *d3plot* generati una volta completate le analisi, è possibile osservarne i risultati: in Figura 5.15 sono riportati tre differenti frame di una simulazione presa come esempio, con i *contour plot* della distribuzione degli sforzi assiali per le travi del campione lattice.

Nel primo frame è presente la cella prima dell'inizio del test (t = 0). Iniziata la simulazione, si può osservare come, durante la compressione della struttura, le beam orizzontali assumano uno stato di trazione (raffigurate in blu), mentre quelle posizionate a 45° subiscano stress di compressione (tonalità sul rosso), come intuitivamente prevedibile a causa della connessione fra le travi.

Dopo aver raggiunto il picco di forza rappresentato nel secondo frame (dopo 1.7 ms di simulazione), c'è un continuo abbassamento generale dei valori di stress dovuto alla rottura di un alto numero di elementi 1D, fino a raggiungere il frame finale (terza immagine in Figura 5.15).

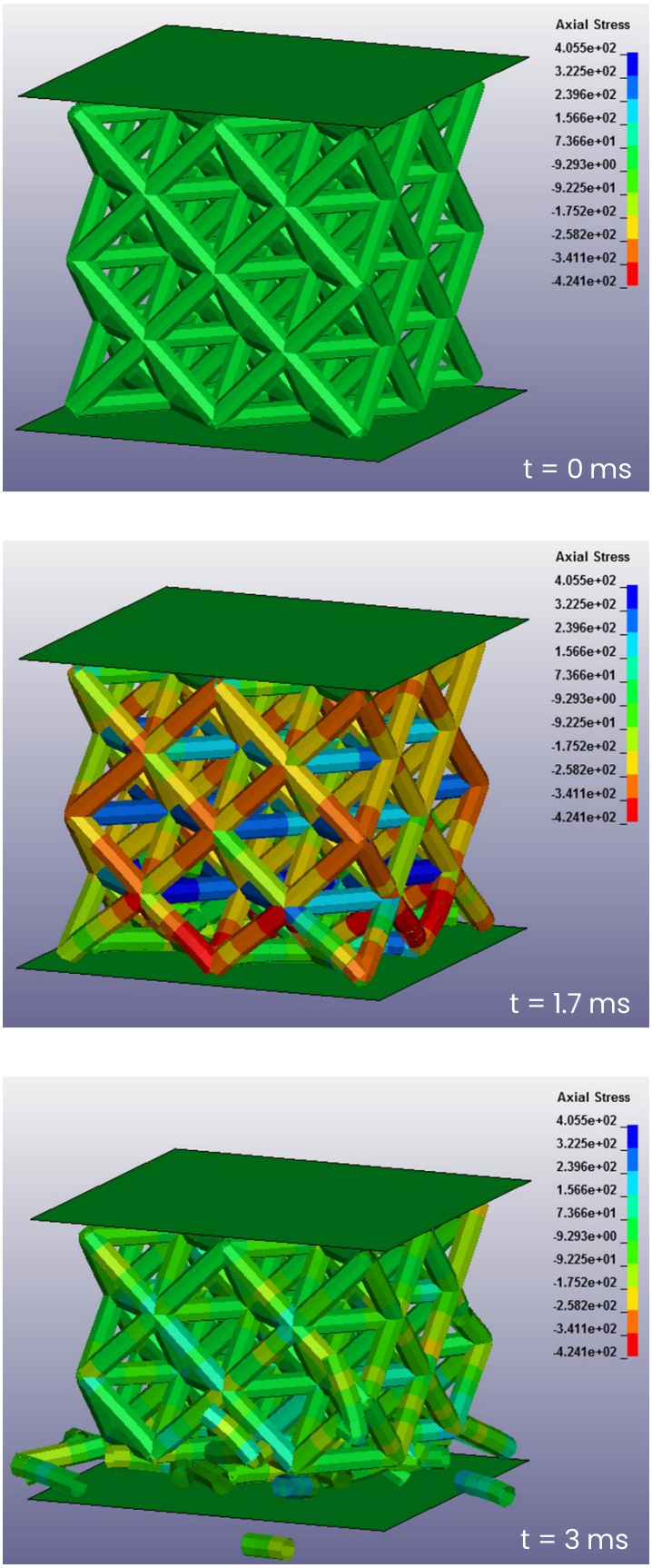

Figura 5. 15 - Tre frame consecutivi della simulazione in LS-Dyna con contour plot raffigurante lo stress assiale delle beam

Quindi, sono state condotte varie simulazioni numeriche, ogni volta generando in maniera casuale, con MATLAB, i file "*material.k*", "*element_beam.k*" e "*part.k*", come spiegato nel paragrafo precedente.

In particolare, lo script per la generazione dei file *.k* e le successive simulazioni FEA sono stati lanciati per 30 volte, ottenendo quindi 30 differenti curve forza-spostamento (Figura 5.16). In questo modo, mediante questa serie di test con associazioni casuali fra le material card e le beam del modello, si tiene conto delle diverse difettosità, interne e superficiali che possono presentarsi nella struttura, influenzando significativamente la risposta a compressione della cella lattice.

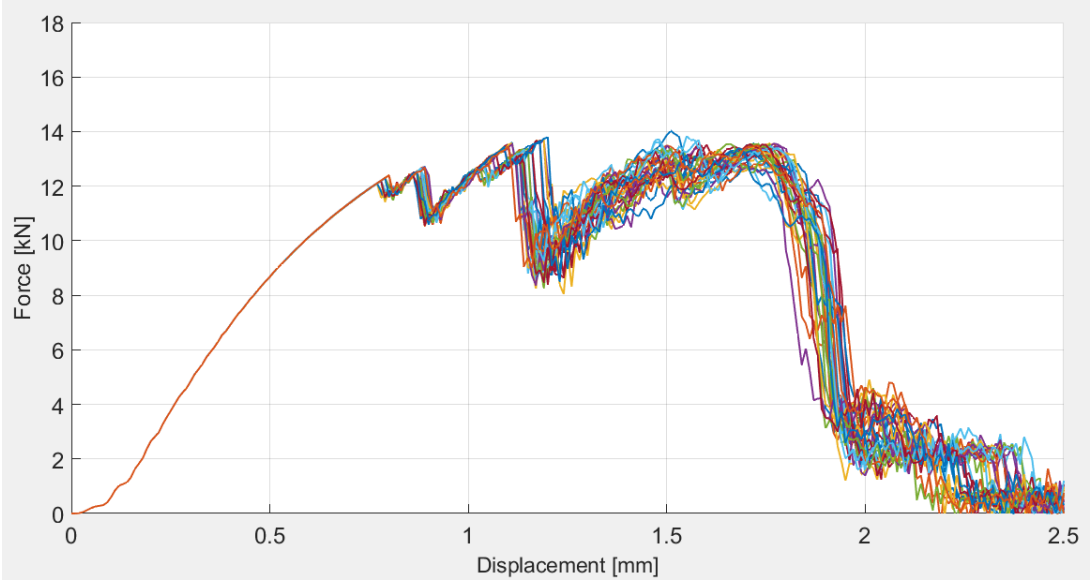

Figura 5. 16 - Famiglia di curve forza-spostamento estratte dalle 30 simulazioni FEA eseguite su LS-Dyna utilizzando le curve ottenute dal piano sperimentale

Determinata questa famiglia di curve, è stato possibile definire un intervallo di variabilità per le risposte numeriche ottenute, al fine di confrontarle con i risultati sperimentali del test di compressione quasi-statico sul provino.

Questo confronto è visibile in Figura 5.17, in cui è presente la curva sperimentale per la cella 2x2, la banda di dispersione data dai risultati delle simulazioni su LS-Dyna e il suo andamento medio. L'intervallo di variabilità è dato da una curva limite inferiore e da una superiore, definite rispettivamente dai valori minimi e massimi della forza in ogni istante fra tutti i risultati ottenuti dalle analisi agli elementi finiti.

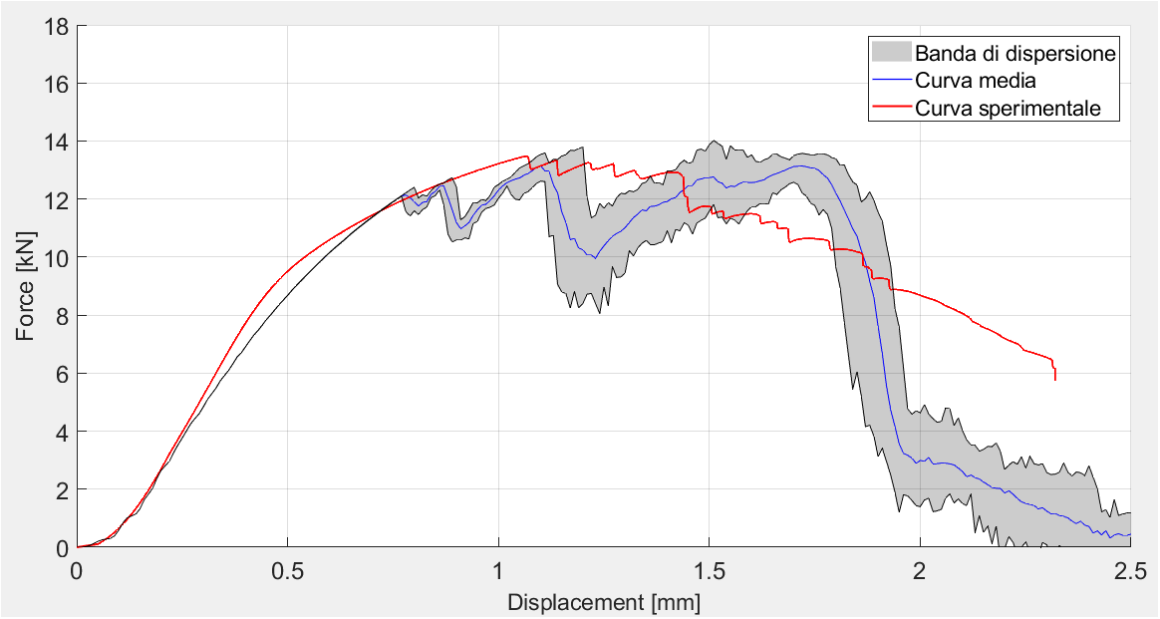

Figura 5. 17 - Confronto fra la curva sperimentale e la banda di dispersione ottenuta dalle analisi numeriche

Dal grafico si nota come la banda di dispersione ottenuta sembri approssimare abbastanza bene la curva sperimentale; in particolare, si ottengono dei valori molto simili per quanto riguarda tratto elastico, forza massima ed energia assorbita (informazione fornita dall'area sottesa alla curva forza-spostamento).

Quindi, sebbene si osservi un andamento più segmentato per le curve ottenute dalle simulazioni numeriche (probabilmente associato ad alcuni parametri incogniti nella simulazione della struttura lattice in LS-Dyna, come l'attrito fra le pareti e il provino), queste risultano approssimare in maniera abbastanza soddisfacente il comportamento sperimentale per la cella 2x2 analizzata, convalidando le analisi microscala eseguite mediante Abaqus CAE.

Inoltre, l'analisi multiscala e l'inserimento di beam con difettosità casuale, permette di ottenere una serie di curve da cui è possibile estrarre, mediante opportune considerazioni statistiche, il caso peggiore, dato dalla curva limite inferiore. Quest'ultima può essere utilizzata nella progettazione e nell'analisi agli elementi finiti di celle lattice future evitando di inserire difettosità effettive all'interno del modello, ma utilizzando un andamento conservativo per la definizione delle sue caratteristiche meccaniche. In Figura 5.18 è presente il confronto fra la curva sperimentale e la curva limite inferiore dalla banda di dispersione.

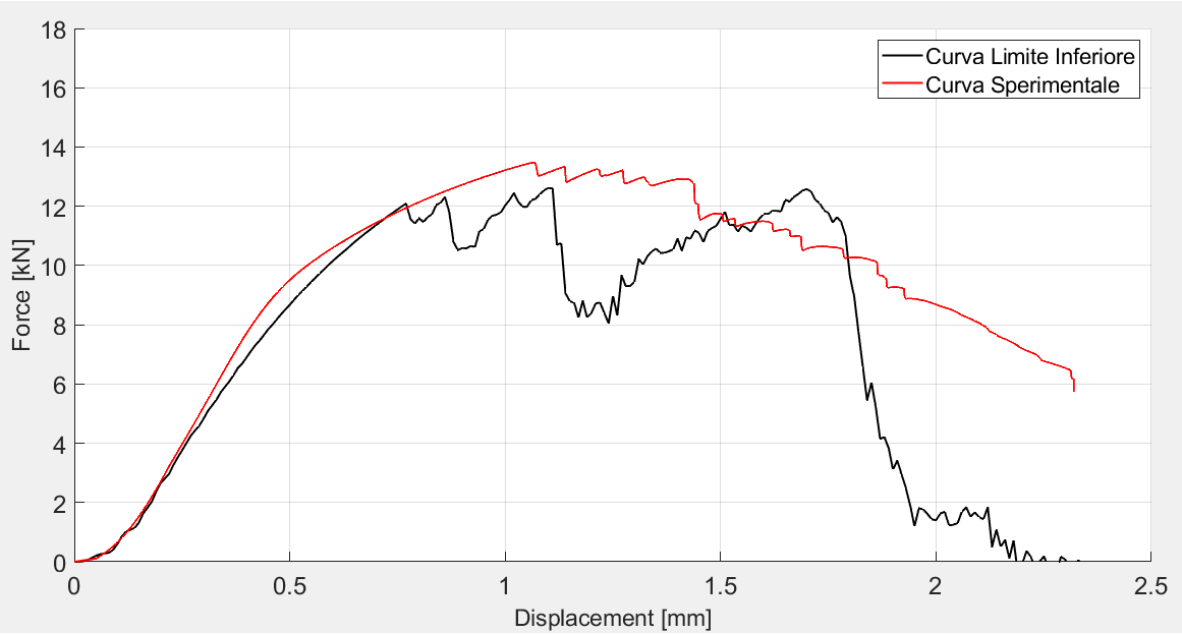

Figura 5. 18 - Confronto fra la curva sperimentale e la curva limite inferiore della banda di dispersione ottenuta

6. Algoritmo machine learning

Dopo aver validato i risultati delle simulazioni FEA per il piano sperimentale con la risposta empirica mediante il confronto in macroscala, è possibile utilizzare le curve omogeneizzate del database per addestrare l'algoritmo di Machine Learning. Tale modello ML, come già anticipato nei capitoli precedenti, avrà lo scopo di prevedere le caratteristiche meccaniche per gli RVE della struttura lattice, sostituendo le lunghe e dispendiose analisi agli elementi finiti su Abaqus CAE (analisi microscala).

Tale modello dovrà poi essere convalidato da nuove simulazioni FEA sulla cella lattice, in cui verranno inserite le beam con gli andamenti predetti. Infatti, per confermare l'algoritmo di ML, le curve forza-spostamento ottenute dovranno approssimare bene il comportamento sperimentale, così come fatto dalle strutture composte dagli RVE analizzati mediante Abaqus.

6.1 Addestramento algoritmo ML

Per l'addestramento del modello è stato scritto un codice python da eseguire mediante Google Colab; quest'ultimo è particolarmente utile per lo sviluppo di modelli di machine learning, perché offre un ambiente di sviluppo basato su Jupyter Notebook che consente di scrivere e lanciare blocchi di codice e visualizzarne i risultati in modo interattivo. Inoltre, per l'esecuzione del codice, sono state importate le librerie TensorFlow e Keras, molto utili nello sviluppo di reti neurali.

Dopo aver importato tutte le librerie necessarie nel codice, si caricano i valori di input e output utili all'addestramento, basati sul database descritto nel capitolo precedente.

Gli input sono rappresentati dai 5 parametri di difettosità utilizzati per la definizione del piano sperimentale: dimensione, aspect ratio e orientamento per il difetto interno, mentre ampiezza e periodo per la rugosità superficiale. Le 1701 combinazioni dei valori di tali parametri vengono inserite in un file .csv, che viene quindi caricato nel codice python.

In Figura 6.1 è presente un estratto del file di input in cui sono visibili le prime 9 combinazioni, corrispondenti alla stessa porosità interna (prima classe di difetto orizzontale con AR=1), ma variando i parametri di ampiezza e periodo della rugosità superficiale.

	A	B	C	D	E
	matrix5input				
	Dmax	aspect_ratio	angle	amplitude_rug	period_rug
	Number	Number	Number	Number	Number
1	Dmax	aspect_ratio	angle	amplitude_r...	period_rug
2	0.062	1	0	11.6	125
3	0.062	1	0	11.6	200
4	0.062	1	0	11.6	250
5	0.062	1	0	27.75	125
6	0.062	1	0	27.75	200
7	0.062	1	0	27.75	250
8	0.062	1	0	55.4	125
9	0.062	1	0	55.4	200
10	0.062	1	0	55.4	250

Figura 6. 1 - Estratto del file di input con le prime 9 combinazioni del database

Riguardo gli output, per semplificare l'addestramento dell'algoritmo ed evitare un modello eccessivamente complesso, si è scelto di non utilizzare interamente le curve omogeneizzate, ma solo due punti fondamentali dell'andamento stress-strain, ovvero il punto di fine tratto elastico e il punto di rottura. Infatti, mediante questi due elementi, sarà possibile ricreare la curva omogeneizzata tracciando prima il tratto elastico iniziale e poi la curva plastica, fino a raggiungere la rottura del componente (tale procedimento sarà spiegato più nel dettaglio nei paragrafi successivi).

Per ottenere i due punti, nel modello ML vengono definiti quattro output, ovvero i valori di tensione e deformazione relativi sia alla fine del tratto elastico e sia alla rottura dell'RVE. Analogamente a quanto fatto per gli input, anche gli output vengono caricati mediante un file .csv, in cui vi è una corrispondenza fra le righe del file di input e le rispettive righe del file di output. In Figura 6.2 è presente un estratto del file .csv di output, in cui sono presenti i valori corrispondenti alle prime 9 combinazioni di input già viste in Figura 6.1.

	A	B	C	D
	output2points			
	Stress_Elastico	Strain_Elastico	Stress_Rottura	Strain_Rottura
	Number	Number	Number	Number
1	Stress_Elasti...	Strain_Elasti...	Stress_Rott...	Strain_Rott...
2	147.06	0.26528	404.55	7.527
3	147.34	0.26549	404.95	7.4073
4	147.51	0.26559	405.18	7.2848
5	142.72	0.25788	401.47	7.6223
6	143.54	0.25924	402.63	7.533
7	143.97	0.25997	402.67	7.4222
8	134.17	0.24239	387.71	7.3138
9	134.99	0.24371	390.97	7.2572
10	136.57	0.24662	397.63	7.2736

Figura 6. 2 - Estratto del file di output per le prime 9 combinazioni dei valori di input

Dopo aver definito i dati in input e in output per il modello, si passa a suddividerli randomicamente in tre set distinti (addestramento, validazione e test), garantendo che ogni range abbia una proporzione corretta dei dati totali. Il set di addestramento viene usato per il training del modello, quello di validazione per ottimizzare i parametri del modello e prevenire l'overfitting (il modello non vede questi dati durante l'addestramento, ma li usa per valutare i risultati ad ogni epoca), mentre i dati di test sono utili per valutare l'efficienza dell'algoritmo su ulteriori dati non visti dal modello durante il training, fornendo una stima imparziale delle sue prestazioni.

Questo passaggio viene effettuato mediante le seguenti due linee di codice python:

```
# Divisione dei dati in set di addestramento e test
train_data, test_data, train_labels, test_labels =
train_test_split(input_data_reorganized, output_data_reorganized, test_size=0.2,
random_state=42)

# Divisione dei dati in set di addestramento e validazione
train_data, val_data, train_labels, val_labels = train_test_split(train_data,
train_labels, test_size=0.2, random_state=42)
```

Vi è una prima divisione per cui l'80% dei dati viene utilizzato per l'addestramento (train) e il restante 20% per il test.

Poi, con la seconda suddivisione, il 20% dei dati di addestramento restanti è usato come set di validazione; quindi, poiché il set di addestramento originale corrispondeva all'80% dei dati totali, il set di validazione sarà il 20% di questo 80%, ovvero il 16% del database intero, lasciando il restante 64% per l'addestramento effettivo.

Inoltre, col comando '`random_state=42`' i dati da inserire nei diversi set sono selezionati casualmente, ma in maniera riproducibile, ottenendo la stessa divisione ogni volta che viene eseguito il codice di addestramento.

In seguito, si passa alla normalizzazione dei dati di input mediante le linee di codice in Figura 6.3, utilizzando la funzione '*StandardScaler()*' dalla libreria Keras; tale processo viene utilizzato per standardizzare i dati affinché abbiano complessivamente una media pari a 0 e varianza pari a 1. Questa tecnica è molto importante per la definizione di reti neurali, perché aiuta a velocizzare la convergenza durante l'addestramento e le prestazioni generali del modello. I dati di output, invece, non vengono normalizzati, al fine di mantenere intatto il loro significato fisico e rendendo più semplice l'interpretazione dei risultati.

```
# Normalizzazione dei dati di input
scaler = StandardScaler()
train_data = scaler.fit_transform(train_data)
val_data = scaler.transform(val_data)
test_data = scaler.transform(test_data)
```

Figura 6. 3 - Normalizzazione dei dati di input con python

Come già descritto nel capitolo introduttivo di questo lavoro di tesi, è necessaria una funzione di perdita che misuri le performance del modello durante ogni epoca. Nello script è stata definita una *loss function* personalizzata per assegnare pesi diversi agli errori quadratici medi (MSE) calcolati sui diversi output, dando maggiore importanza ad alcune componenti rispetto ad altre. In particolare, dalla Figura 6.4 si può evincere come sia stata data maggiore importanza ai due parametri di deformazione (soprattutto a quella di fine tratto elastico), poiché, avendo dei valori numerici nettamente più bassi rispetto a quelli degli stress, è più difficile prevederli in maniera accurata.

```python
def weighted_mse(y_true, y_pred):
    # Pesi: [stress elastico, strain elastico, stress rottura, strain rottura]
    weights = np.array([1.0, 5.0, 1.0, 3.0])
    squared_diff = K.square(y_true - y_pred)
    weighted_squared_diff = squared_diff * weights
    return K.mean(weighted_squared_diff, axis=-1)
```

Figura 6. 4 – Definizione in python della funzione di perdita personalizzata per assegnare pesi diversi agli MSE sui diversi output

Dunque, si passa a definire la struttura della rete neurale, usando le linee di codice in Figura 6.5. Dopo aver effettuato diverse prove variando i parametri caratterizzanti il training (già descritti nel Paragrafo 1.3), tale struttura è risultata essere la migliore in termini di costi computazionali e precisione degli output finali. Si nota come, oltre allo strato di input (5 nodi corrispondenti ai 5 parametri di input) e a quello di output (4 nodi corrispondenti ai 4 parametri di output), sono presenti 6 strati nascosti, con rispettivamente 256, 128, 64, 32, 16 e 8 neuroni al loro interno. In particolare, tutti i *layers* della rete sono strati "densi", ovvero in cui ogni neurone riceve input da tutti i neuroni del precedente strato, con funzione di attivazione ReLU (Rectified Linear Unit).

```python
# Definizione del modello
model = Sequential([
    Dense(256, activation='relu', input_shape=(5,)),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
    Dense(8, activation='relu'),
    Dense(4)
])
```

Figura 6. 5 – Struttura della rete neurale per l'addestramento del modello ML

Prima di eseguire l'addestramento, vengono inserite due funzioni di *callback* presenti in Keras: '*ModelCheckpoint*' e '*EarlyStopping*' (attraverso il codice in Figura 6.6). Il primo salva il modello ogni volta che si ha un miglioramento sulla perdita dei dati di validazione, mentre il secondo interrompe l'addestramento se non ci sono riduzioni significative dell'errore quadratico medio (`min_delta=0.001`) per un certo numero di epoche (`patience=1000`).

Così facendo, si conserva il modello migliore fra quelli ottenuti durante il training, evitando di continuare l'addestramento una volta raggiunta la convergenza sulla funzione di perdita.

```
# Callback per ModelCheckpoint
checkpoint_filepath = 'best_model.h5'
checkpoint = ModelCheckpoint(checkpoint_filepath,
                             monitor='val_loss',
                             verbose=1,
                             save_best_only=True,
                             mode='min')

# Callback per l'early stopping
early_stopping = EarlyStopping(monitor='val_loss',
                               min_delta=0.001,
                               patience=1000,
                               verbose=1,
                               restore_best_weights=True)
```

Figura 6. 6 – Inserimento funzioni di callback 'ModelCheckpoint' e 'EarlyStopping'

Quindi, viene compilato e addestrato il modello (Figura 6.7) utilizzando i parametri fondamentali già descritti nel capitolo introduttivo relativo al ML; in particolare, dopo aver provato numerose combinazioni, si è optato per 20000 epoche con un batch size pari a 32 e un learning rate pari a 0.00001. Inoltre, viene utilizzata come funzione di perdita quella definita in precedenza nel codice, basata su un MSE ponderato, mentre come metrica viene usato l'errore assoluto medio (MAE); in generale, le metriche vengono inserite per monitorare e valutare ulteriormente le prestazioni del modello durante l'addestramento, ma non influiscono in modo diretto sull'ottimizzazione dello stesso (a differenza della funzione di perdita).

```
learning_rate = 0.00001
batch_size = 32

# Compilazione del modello
optimizer = Adam(learning_rate=learning_rate)
model.compile(optimizer=optimizer,
              loss=weighted_mse,
              metrics=['mae'])

# Addestramento del modello
history = model.fit(train_data, train_labels, epochs=20000, batch_size=batch_size,
                    validation_data=(val_data, val_labels),
                    callbacks=[checkpoint, early_stopping])
```

Figura 6. 7 - Compilazione e addestramento del modello ML

Infine, il modello viene prima salvato, permettendone il caricamento successivo senza doverlo addestrare nuovamente, e poi ne viene valutata l'efficienza mediante una serie di grafici.

Il primo, presente in Figura 6.8, monitora la riduzione del valore delle perdite proseguendo nelle epoche, mostrando l'andamento del *training loss* e del *validation loss*. Il primo è dato dall'MSE tra i valori predetti dal modello e quelli veri (forniti dal file degli output), valutando la capacità del modello di adattarsi ai dati usati durante il training per ogni epoca. Il secondo fornisce la stessa informazione, ma usando il dataset di validazione, quindi valutando le prestazioni su nuovi dati, non visti durante l'addestramento.

Si può notare come si raggiunga la convergenza e, quindi, la fine del training (valutata mediante il callback dell'*EarlyStopping*) dopo circa 15000 epoche, ottenendo dei valori di perdita abbastanza bassi e soddisfacenti (training loss = 0.039, validation loss = 0.042).

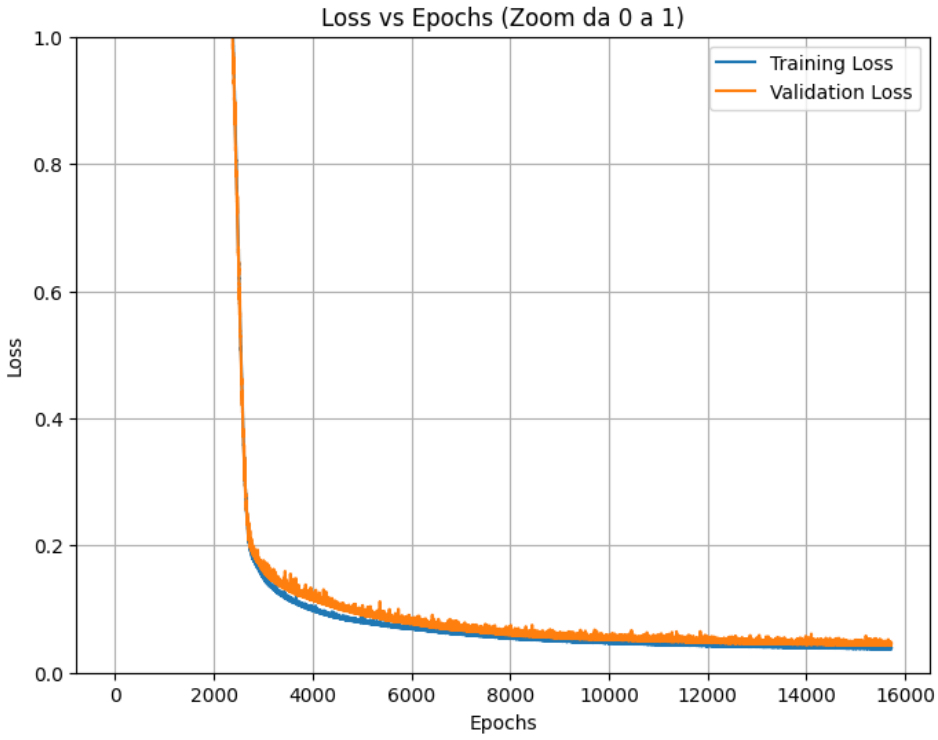

Figura 6. 8 - Andamento del training loss e del validation loss durante le epoche di addestramento

Inoltre, sono stati tracciati quattro grafici a dispersione, uno per ogni output del modello, in cui sono rappresentati sulle ascisse i valori reali dal file di output e sulle ordinate quelli predetti dalla rete su dei dati di test. In questo modo, è possibile rappresentare le predizioni ideali con una linea passante dall'origine e inclinata di 45°, facendo in modo che più un punto sul grafico è vicino a quella linea e più il valore di output predetto è simile a quello reale.

In Figura 6.9 sono presenti i plot relativi a 10 campioni di test, da cui si può subito notare come per gli stress ci sia una predizione quasi perfetta, mentre nelle deformazioni sembra esserci visivamente qualche differenza. Questa discrepanza, però, è dovuta solo allo zoom utilizzato per tali grafici, perché analizzando numericamente i risultati si evince come per gli strain di rottura ci sia, nel caso peggiore, giusto una differenza di 0.1% fra valore predetto e valore reale, mentre la maggiore distanza dall'idealità per la deformazione di fine tratto elastico è solo pari a circa 0.01%.

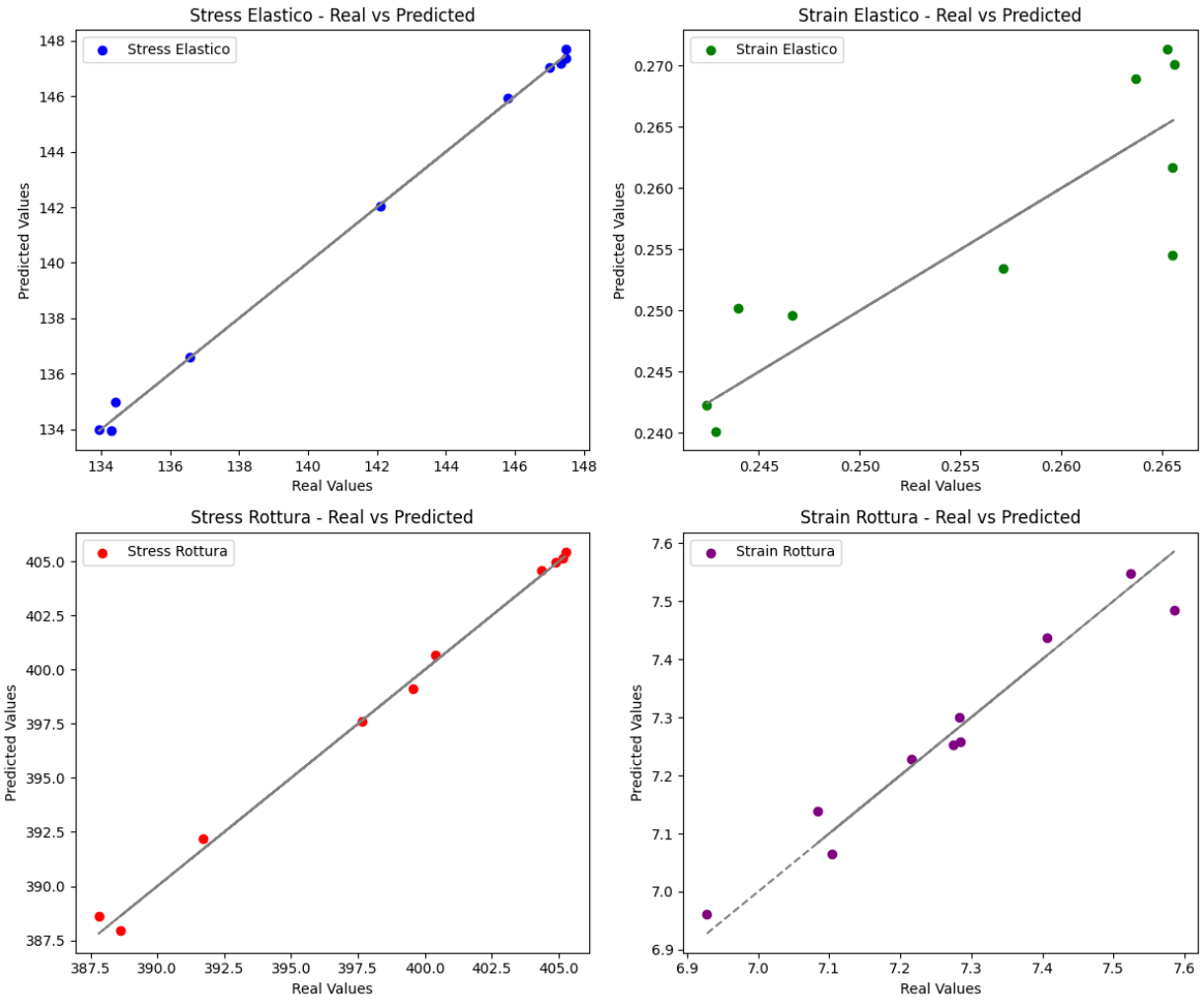

Figura 6. 9 - Grafici a dispersione per valutare l'accuratezza degli output predetti dal modello ML

6.2 Predizione curve omogeneizzate

Dopo aver confermato il corretto addestramento del modello si è passato alla sua validazione, utilizzando gli stessi valori di input usati per l'addestramento e valutando se l'algoritmo riuscisse a prevedere bene le quantità di output per le 1701 combinazioni date dal database definito nei capitoli precedenti. Come per l'addestramento, anche per il caricamento e utilizzo del modello ML viene utilizzato un codice python da eseguire mediante Google Colab.

Dopo aver riscritto la funzione di perdita personalizzata usata per il training (già mostrata in Figura 6.4), è possibile caricare il modello addestrato in formato .h5 (utilizzato solitamente per memorizzare algoritmi di machine learning), mediante la seguente linea di codice in Figura 6.10:

```
# Caricamento del modello addestrato
model = load_model('stress_strain_model.h5', custom_objects={'weighted_mse': weighted_mse})
```

Figura 6. 10 - Caricamento modello ML addestrato

In seguito, viene importato come file di input lo stesso utilizzato per l'addestramento e i dati al suo interno vengono normalizzati alla stessa maniera di quanto fatto per l'addestramento, adattando la funzione '*StandardScaler()*' ai nuovi dati di input (Figura 6.11).

```
# Normalizzazione dei nuovi dati di input
scaler = StandardScaler()
scaler.fit(train_data)  # Adatta lo scaler sui dati di addestramento
normalized_new_input_data = scaler.transform(new_input_data)
```

Figura 6. 11 - Normalizzazione dei nuovi dati di input

Quindi, si passa all'utilizzo del modello addestrato per effettuare le previsioni sui nuovi dati di input normalizzati, mediante la linea di codice python in Figura 6.12.

```
# Previsione utilizzando il modello addestrato
predictions = model.predict(normalized_new_input_data)
```

Figura 6. 12 - Previsione degli output mediante il modello ML

Infine, tutti i dati predetti dal modello vengono inseriti in una tabella, che viene esportata in un file CSV per consentirne un facile utilizzo e visualizzazione (in Figura 6.13 è presente il codice python utilizzato per fare ciò).

```
# Creazione del DataFrame con le predizioni
columns = ['Stress_Elastico', 'Strain_Elastico', 'Stress_Rottura', 'Strain_Rottura']
predictions_df = pd.DataFrame(predictions, columns=columns)

# Salvataggio del DataFrame in un file CSV
predictions_df.to_csv('predicted_curves_2x2.csv', index=False)
```

Figura 6. 13 - Salvataggio output in un file CSV

Dopo pochissimi secondi dall'esecuzione del codice si ottiene il file di output desiderato; in Figura 6.14 è presente un estratto di tale file, in cui sono visibili le prime 9 combinazioni dal database, corrispondenti ad un unico difetto interno (prima classe di difetto orizzontale con AR=1), ma variando i parametri di ampiezza e periodo della rugosità superficiale.

	A	B	C	D
	predictedcurves2x2			
	Stress_Elastico	Strain_Elastico	Stress_Rottura	Strain_Rottura
	Number	Number	Number	Number
1	Stress_Elasti...	Strain_Elasti...	Stress_Rottu...	Strain_Rottu...
2	146.92352	0.26125932	404.66498	7.653421
3	147.13295	0.25138688	404.88153	7.4584746
4	147.35109	0.243994	405.47885	7.364984
5	142.63525	0.24623704	401.32892	7.7287993
6	143.49313	0.2420218	402.49402	7.592279
7	143.70311	0.23403001	402.76834	7.5025115
8	134.09152	0.2416861	387.76587	7.378061
9	134.70908	0.23717332	391.17877	7.29841
10	136.71495	0.24650025	397.7752	7.3356566

Figura 6. 14 - Estratto del file CSV di output

Quindi, mediante il modello addestrato, è stato possibile ottenere come output i quattro valori caratterizzanti i punti di fine tratto elastico e di rottura per le curve omogeneizzate degli RVE. Dopodiché, mediante un codice MATLAB, è possibile tracciare le curve stress-strain per intero grazie a questi due punti fondamentali.

L'idea di base è quella di tracciare inizialmente una linea retta che colleghi l'origine con il punto di fine tratto elastico, per poi collegare quest'ultimo col punto di rottura mediante un andamento curvilineo, il quale ha il suo massimo nel punto finale.

Si inizia importando il file CSV con i due punti fondamentali predetti dal modello per ogni combinazione di input, i quali dovranno essere uniti mediante una curva di Bézier. Questa è una tipologia di curva parametrica utilizzata per tracciare andamenti regolari e continui mediante l'utilizzo di diversi punti di controllo. Nel caso in esame, è stata utilizzata una curva di Bézier cubica, la quale ha quattro punti di controllo (P_0, P_1, P_2 e P_3) ed è definita da una combinazione lineare di questi punti, descritta dall'equazione parametrica (dove t varia fra 0 a 1):

$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

Per utilizzare tale equazione in ambiente MATLAB è stata definita una funzione mediante la quale vengano calcolati e restituiti i punti della curva di Bézier, utilizzando una combinazione lineare dei valori del parametro t e dei coefficienti relativi ai punti di controllo della curva (Figura 6.15). Tale funzione non è specifica per il polinomio cubico del caso in esame, ma può essere utilizzata per equazioni parametriche di qualsiasi grado.

```
function [points] = bezier_curve(coefficients, t_values)
    % Calcolo dei punti della curva di Bézier dati i coefficienti e i valori di t
    n = length(coefficients) - 1;
    points = zeros(size(t_values));

    for k = 0:n
        points = points + coefficients(k+1) * nchoosek(n, k) * t_values.^k .* (1 - t_values).^(n - k);
    end
end
```

Figura 6. 15 – Definizione funzione MATLAB per curva di Bézier di grado generico

In seguito, dopo aver determinato il parametro t con 60 punti equidistanti, sono stati definiti i punti di controllo per ciascuna configurazione di output mediante un ciclo for, come visibile dal codice in Figura 6.16; si può notare come per P_0 e P_3 siano stati usati rispettivamente i punti di fine tratto elastico e di rottura, mentre per P_1 e P_2 si è optato per i valori presenti nello script, dopo vari tentativi per ottimizzare il risultato finale.

Quindi, dopo aver calcolato i coefficienti del polinomio cubico per le componenti x e y attraverso i punti di controllo, è stata utilizzata la funzione definita in precedenza per ottenere i punti della curva di Bézier, la quale, per ciascuna combinazione degli output predetti, collega il punto di fine tratto elastico col punto di rottura.

```
% Preallocazioni per i punti di Bézier
t_values = linspace(0, 1, 60);  % Utilizzo di 60 punti
x_bezier = zeros(length(t_values), num_curves);
y_bezier = zeros(length(t_values), num_curves);

% Calcola i punti di Bézier per ogni curva
for i = 1:num_curves
    P0 = [Strain_Elastico(i); Stress_Elastico(i)];
    P3 = [Strain_Rottura(i); Stress_Rottura(i)];
    P1 = P0 + [1; Stress_Elastico(i)/Strain_Elastico(i)*0.44];
    P2 = P3 - [1; 5];

    % Calcola i coefficienti per la curva di Bézier con 60 punti
    coefficients_x = [P0(1) P1(1) P2(1) P3(1)];
    coefficients_y = [P0(2) P1(2) P2(2) P3(2)];

    % Calcola i punti della curva di Bézier
    x_bezier(:, i) = bezier_curve(coefficients_x, t_values);
    y_bezier(:, i) = bezier_curve(coefficients_y, t_values);
```

Figura 6. 16 – Definizione delle curve di Bézier cubiche per ciascuna configurazione di output

Infine, ciascuna curva di Bézier ottenuta viene salvata in un file .txt, aggiungendo l'origine degli assi (0;0) come prima riga di ogni file. In questo modo si ottengono una serie di curve caratterizzate da due parti principali: un primo tratto elastico rettilineo fra l'origine degli assi e il punto di fine tratto elastico, la cui pendenza mi determina il modulo elastico del materiale, e un secondo tratto curvilineo, che arriva al punto di rottura finale (con tensione e deformazione massima) in maniera circa tangente all'orizzontale, come gran parte delle curve ottenute dalle simulazioni su Abaqus.

In Figura 6.17 sono raggruppate tutte le 1701 curve tracciate nel modo appena descritto mediante i due punti fondamentali predetti dal modello ML. In particolare, e come auspicabile, si nota subito una certa somiglianza con le curve del database iniziale già mostrate in Figura 5.2.

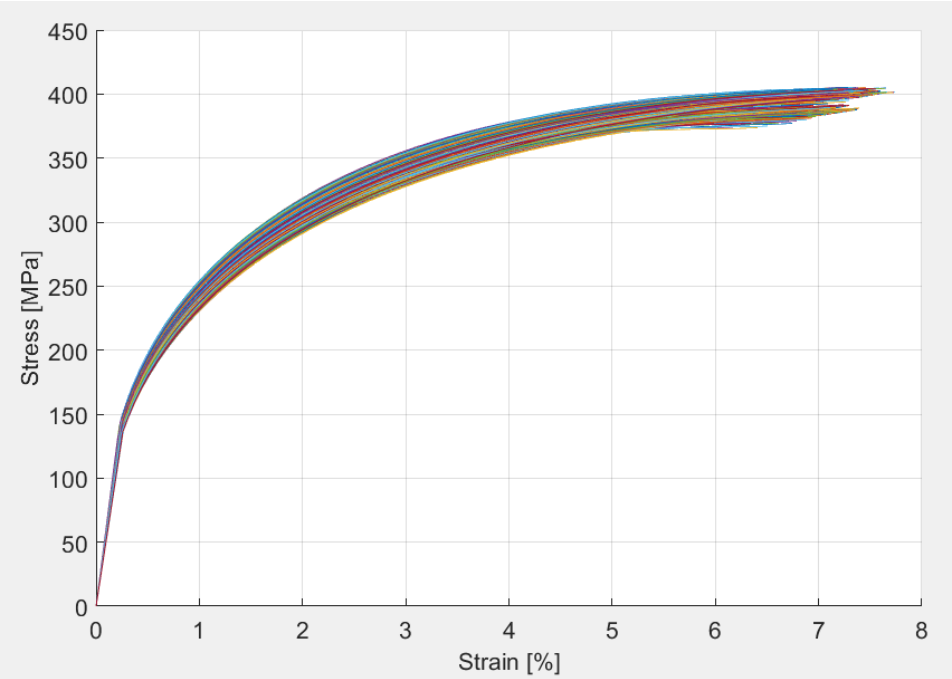

Figura 6. 17 - Insieme di tutte le curve predette

Infatti, entrando più nel dettaglio, i risultati ottenuti approssimano molto bene le curve omogeneizzate ricavate dalle simulazioni FEA sulle diverse tipologie di RVE. Per l'esempio in Figura 6.18 è stata presa casualmente la 1697° curva da entrambi i database, corrispondente ai seguenti input: D_{max}= 631 µm, AR = 2, difetto verticale, ampiezza rugosità = 27.75 µm, periodo rugosità = 200 µm.

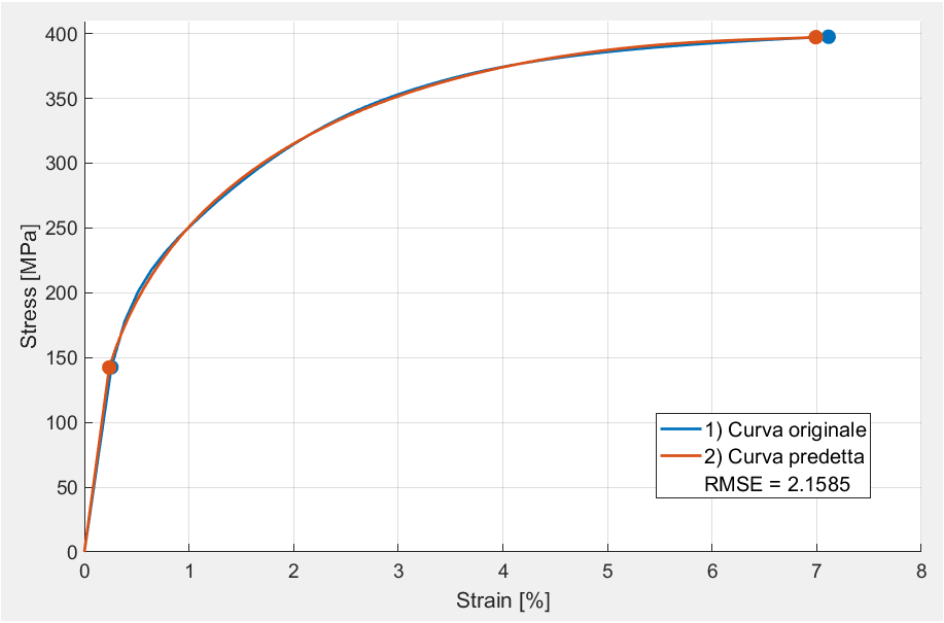

Figura 6. 18 - Esempio di confronto fra curva originale e curva predetta

Da tale esempio si evince perfettamente come ci sia una buonissima approssimazione (valutata da un errore quadratico medio pari a solo 2.16) fra la curva originale fornita dalle simulazioni su Abaqus e la curva predetta mediante il modello di Machine Learning.

In conclusione, una cosa fondamentale da puntualizzare è che le simulazioni agli elementi finiti per definire il piano sperimentale hanno impiegato giorni interi per poter essere completate, mentre con le predizioni mediante l'algoritmo di ML sono stati necessari pochissimi secondi per avere dei risultati che, come visibile dai grafici appena mostrati, possono tranquillamente sostituire le dispendiose analisi da eseguire con Abaqus.

6.3 Validazione modello ML

Per validare definitivamente il modello di Machine Learning, le curve predette sono state utilizzate analogamente a quanto fatto nel Capitolo 5 col database ricavato mediante le simulazioni su Abaqus CAE.

Infatti, utilizzando il software LS-Dyna, la cella lattice 2x2 è stata discretizzata nuovamente con una mesh di elementi 1D, ma in questo caso, alle beam costituenti la struttura, sono state assegnate casualmente le curve omogeneizzate predette dall'algoritmo di ML. Quindi, mediante questo approccio multiscala, la difettosità della singola trave viene valutata su scala locale mediante il Machine Learning e la risposta ottenuta viene associata alla struttura lattice globale.

Con LS-Dyna viene simulata nuovamente la risposta a compressione quasi-statica della cella 2x2, utilizzando lo stesso codice descritto nel Capitolo 5 per la definizione dei file keyword. Infatti, l'unica cosa che è stata modificata è la tabella contenente le 1701 curve omogeneizzate da assegnare casualmente agli elementi 1D della struttura; in questo modo, l'unico file che subirà una variazione significativa sarà il "*material.k*", che in questo caso sarà caratterizzato dagli andamenti ottenuti col modello ML e già mostrati in Figura 6.17.

Quindi, sia tutta la parte relativa all'assegnazione casuale delle material card alle travi della struttura (ma con una certa probabilità definita in base alla frequenza sperimentale di ciascuna classe di difetto) e sia il file keyword principale da lanciare su LS-Dyna con tutte le caratteristiche delle simulazioni (compresa la legge di spostamento lineare della parete superiore) sono identiche a quanto mostrato nel Capitolo 5, e pertanto non verranno ripetute in questo paragrafo.

Dopo aver generato il modello per LS-Dyna, anche in questo caso sono state condotte 30 simulazioni numeriche FEA, ogni volta generando in maniera casuale, con MATLAB, i file "*material.k*", "*element_beam.k*" e "*part.k*", e ottenendo nuovamente 30 differenti curve forza-spostamento per la risposta a compressione quasi-statica della cella 2x2 (Figura 6.19).

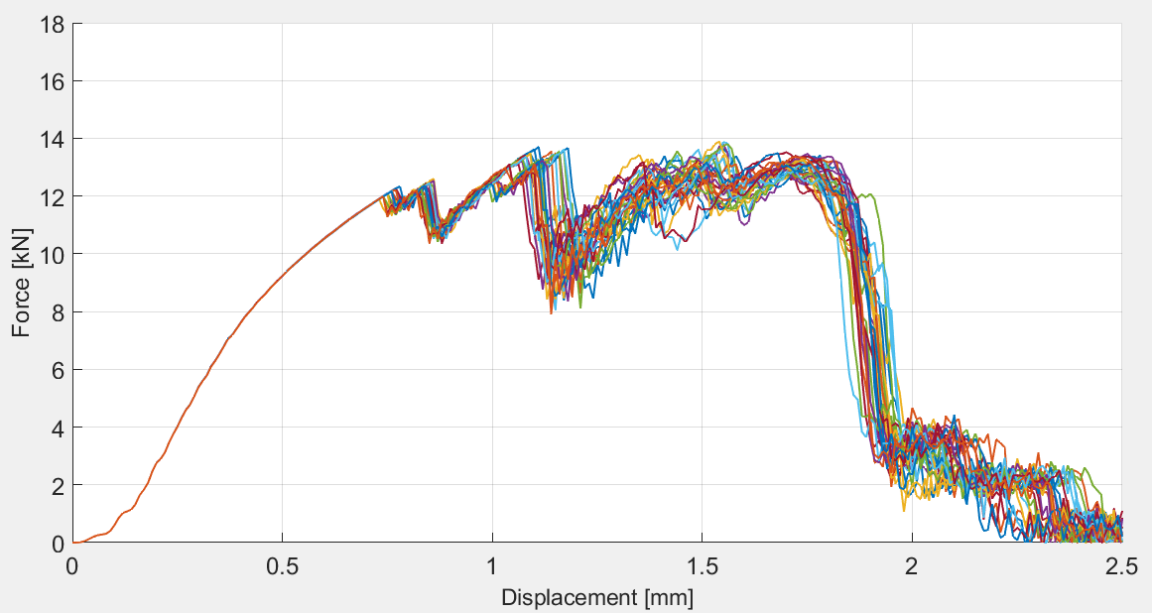

Figura 6. 19 - Famiglia di curve forza-spostamento estratte dalle 30 simulazioni FEA eseguite su LS-Dyna utilizzando le curve ottenute dal modello ML

Dopo aver definito anche per queste analisi una banda di dispersione data dai valori minimi e massimi della forza in ogni istante, è stata possibile compararla con l'intervallo di variabilità ottenuto con gli RVE definiti su Abaqus. Il confronto è visibile in Figura 6.20, da cui si evince una grande somiglianza fra i due fasci di curve, confermando definitivamente la possibilità di sostituire le simulazioni mediante Abaqus con gli output ottenuti dal modello ML addestrato.

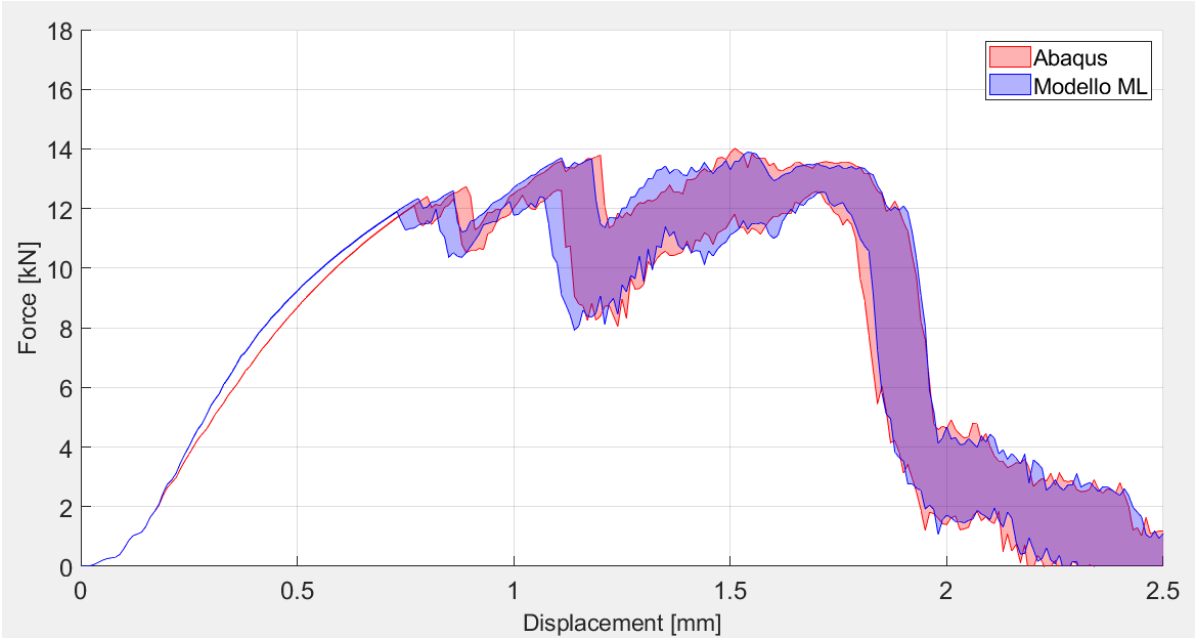

Figura 6. 20 – Confronto fra le bande di dispersione ottenute dai risultati numerici, una per il modello con gli RVE simulati con Abaqus (rosso) e una con le curve predette dal modello ML (blu)

Dopodiché, col grafico tracciato in Figura 6.21, è possibile confrontare la nuova risposta a compressione numerica con quella sperimentale del componente (già mostrata in Figura 5.12).

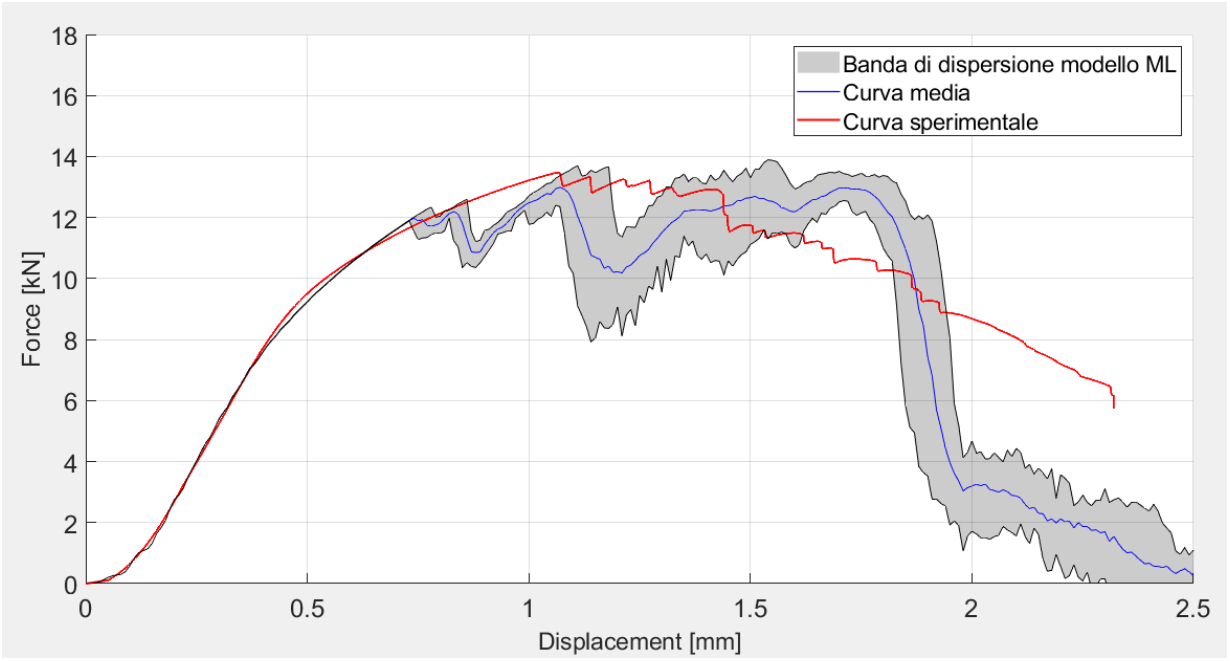

Figura 6. 21 - Confronto fra la curva sperimentale e la banda di dispersione ottenuta grazie al modello ML

Si nota come la banda di dispersione ottenuta approssimi bene la curva sperimentale, come prevedibile data la somiglianza col caso numerico precedente; in particolare, si ottengono valori molto simili per quanto riguarda tratto elastico, forza massima ed energia assorbita, convalidando anche questa simulazione numerica in cui per le analisi microscala viene usato un algoritmo addestrato di machine learning.

Inoltre, un vantaggio fornito dalla analisi multiscala, e quindi dalla definizione di un intervallo di variabilità, è la possibilità di estrarre il caso peggiore (dato dalla curva limite inferiore) al fine di utilizzarlo per la progettazione di celle lattice future, come già descritto in precedenza. In Figura 6.22 è presente il confronto fra la curva sperimentale e le curve limite inferiori estratte dalla nuova banda di dispersione e dall'intervallo di variabilità ottenuto dall'analisi multiscala. Si nota come i due andamenti numerici siano quasi sovrapposti, confermando ulteriormente le ottime previsioni del modello di machine learning e fornendo in entrambi i casi una curva di progetto abbastanza conservativa, ma comunque rappresentative della prova empirica.

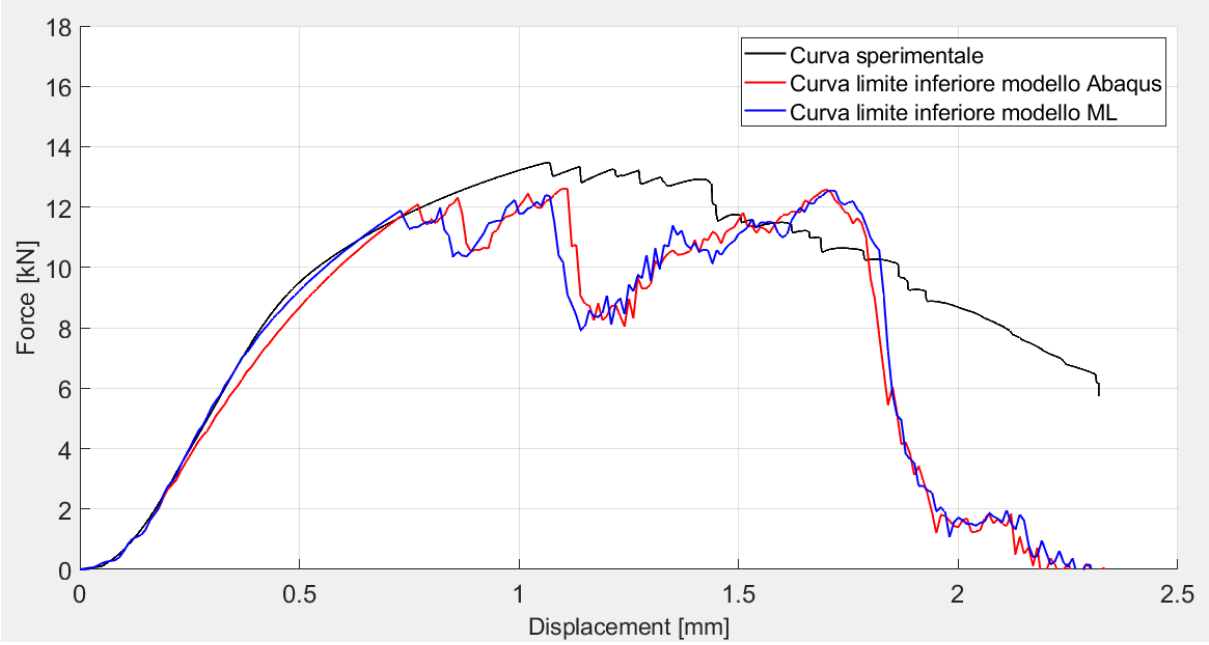

Figura 6. 22 - Confronto fra la curva sperimentale e le curve limite inferiori delle due bande di dispersione ottenute

7. Conclusioni

In conclusione, questo lavoro di tesi ha permesso di definire un modello numerico agli elementi finiti e un algoritmo di machine learning per simulare e prevedere la risposta a compressione di celle lattice prodotte in Additive Manufacturing.

Da prove sperimentali eseguite precedentemente, sono stati ripresi valori relativi a caratteristiche geometriche e meccaniche, oltre che parametri di difettosità interna e superficiale. Sono state impiegate queste informazioni per creare il modello agli elementi finiti della struttura lattice utilizzando un approccio multiscala, quindi partendo dalla definizione di un elemento di volume rappresentativo su scala locale (elemento beam della struttura). A tale modello è stato assegnato un difetto interno con una geometria ellissoidale dipendente da volume, diametro massimo e aspect ratio, ricavabili da scansioni di micro-tomografia computerizzata su componenti reali. Inoltre, la rugosità superficiale è stata simulata con un profilo sinusoidale che, mediante il confronto con la profilometria estratta da analisi con il rugosimetro su un pezzo as-built in AlSi10Mg, è risultata essere una buona approssimazione di rugosità superficiali realistiche.

Si è quindi passato alle simulazioni agli elementi finiti su tali RVE con il software Abaqus CAE, che hanno evidenziato come la caratteristica principale da tenere in considerazione sia la sezione resistente della beam sottoposta a trazione. A causa di tale caratteristica, dalle simulazioni FEA è stato possibile definire cinque parametri di difettosità principali che influenzano maggiormente la risposta meccanica della beam: tre relativi al difetto interno (dimensione, orientamento e aspect ratio) e due riguardanti la rugosità superficiale (ampiezza e periodo del profilo sinusoidale). Inoltre, dalle simulazioni sono state estratte le curve omogeneizzate tensione-deformazione per gli RVE, le quali definiscono le caratteristiche meccaniche della beam in esame sulla base delle sue difettosità. Dunque, è stato definito un ampio database contenente le curve omogeneizzate di varie tipologie di beam, caratterizzate da tutte le possibili combinazioni dei parametri analizzati con Abaqus, relativi a difetti interni e superficiali.

Con tale database sono state definite le material card da utilizzare per le analisi agli elementi finiti di una cella lattice 2x2 (macroscala) mediante il software LS-Dyna, associando casualmente (mediante uno script MATLAB) le curve omogeneizzate agli elementi 1D costituenti il modello numerico della struttura intera.

Le simulazioni FEA sul modello hanno generato delle curve forza-spostamento caratteristiche della cella lattice, definendo una banda di dispersione per tali risultati. Tale intervallo di variabilità è stato poi confrontato con il comportamento sperimentale, evidenziando come il modello numerico approssimi in maniera abbastanza soddisfacente i risultati del test empirico, soprattutto in termini di forza massima, tratto elastico ed energia assorbita. In questo modo, inoltre, è stato validato il database generato con i risultati delle simulazioni eseguite sugli RVE mediante Abaqus.

Tale database è stato utilizzato nuovamente, ma per addestrare un algoritmo di machine learning per prevedere il carico e la deformazione al termine del tratto elastico e nel punto di rottura sulle curve omogeneizzate degli RVE (ottenendo quindi quattro valori di output), fornendo come input i cinque parametri di difettosità più influenti già descritti. Mediante l'ausilio di una curva parametrica di Bézier è stato possibile ricreare le curve omogeneizzate partendo dai due punti di output forniti dal modello ML.

Dopo aver effettuato il training dell'algoritmo ML, questo è stato utilizzato per prevedere le curve omogeneizzate degli RVE sulla base degli stessi input usati per addestrare il modello. Per validare i risultati ottenuti, questi sono stati presi come database per definire nuovamente le material card del modello numerico di cella 2x2 su LS-Dyna, analogamente a quanto fatto in precedenza. Anche in questo caso la banda di dispersione ottenuta sul diagramma forza-spostamento è risultata essere una buona approssimazione della curva ottenuta dai test sperimentali.

Inoltre, è risultata essere limitata la differenza fra la banda di dispersione ottenuta utilizzando le curve predette e l'intervallo di variabilità dato dalle simulazioni FEA con Abaqus. In particolare, se quest'ultime hanno impiegato diversi giorni di calcolo per essere completate, le curve prodotte dal modello ML, invece, sono state definite in un tempo limitato (inferiore al minuto), garantendo un risparmio significativo in termini di tempo e costi computazionali.

In questo modo è stato raggiunto con successo lo scopo principale di questo lavoro di tesi, ovvero la definizione di un algoritmo di Machine Learning che preveda correttamente la risposta a compressione di una cella lattice prodotta in Additive Manufacturing, e che riesca a sostituire le lunghe e dispendiose analisi FE sugli elementi microscala della struttura lattice in esame.

Studi futuri potranno riguardare il modello numerico multiscala, ad esempio valutando nuovi approcci per definire la rugosità superficiale dell'RVE, oppure stimando l'influenza del diametro della beam stessa.

Inoltre, in questo lavoro di tesi, il modello di machine learning è stato valutato con gli stessi input utilizzati per il suo addestramento, mentre in futuro potrebbe essere ottimizzato per nuovi parametri di input, ad esempio inserendo caratteristiche di difetto riprese da nuove scansioni Micro-CT su differenti tipologie di struttura lattice. Infatti, un ulteriore studio potrebbe essere relativo a celle 3x3 (30 x 30 x 30 mm) oppure a geometrie reticolari differenti da quella analizzata nei capitoli precedenti.

In questo modo, le enormi potenzialità offerte dagli algoritmi di machine learning potranno essere sfruttate anche in futuro per ottenere significative riduzioni di tempi e costi per l'esecuzione delle analisi agli elementi finiti.

Appendice

Estratto del codice Python per eseguire le simulazioni (per il D.O.E.) su Abaqus CAE con AR=1; tale estratto è relativo solo al ciclo for annidato (quindi successivo alla definizione delle funzioni ‘generate_cylindrical_rve_ellips’, ‘assign_material’, ‘apply_bcs’ e ‘run_job’):

```
# Memorizza la directory corrente
current_directory = os.getcwd()
# Lettura dei file con le classi di difetto
dmax_values = np.loadtxt('dmax_defects_2x2.txt')
dmin_values_AR1 = np.loadtxt('dmin_aspect_ratio_1_2x2.txt')
# Assegna parametri comuni
size_values = [1.4] # mm
repetitions_angles = 3
angles = [0.0, 45, 90]
repetitions_rug = 3
rug = [11.6/1000, 27.75/1000, 55.4/1000] #/1000 per micron-->mm
repetitions_period = 3
period = [0.125, 0.2, 0.25] #mm
# Cambio directory per entrare nella cartella desiderata
output_directory='beam14x20_AR1'
if not os.path.isdir(output_directory):
    os.mkdir(output_directory)
os.chdir(output_directory)
for i, Dmax in enumerate(dmax_values):
        print("Generating defect with size: {}".format(Dmax))
        Dmin = dmin_values_AR1[i]
        if not os.path.isdir('defect_{}'.format(int(Dmax*100))):
                os.mkdir('defect_{}'.format(int(Dmax*100)))
        os.chdir('defect_{}'.format(int(Dmax*100)))
```

```
        for size in size_values:
                for rep1 in range(repetitions_angles):
                        angle = angles[rep1]
                        model_name1='analysis_{}_{}_{}'.format(int(Dmax*100), int(size*100),
int(angle))
                        if not os.path.isdir(model_name1):
                                os.mkdir(model_name1)
                        os.chdir(model_name1)


                        for rep2 in range(repetitions_rug):
                                rug_value = rug[rep2]
                                model_name2='analysis_{}_{}_{}_{}'.format(int(Dmax*100),
int(size*100), int(angle), int(rug_value*1000))
                                if not os.path.isdir(model_name2):
                                        os.mkdir(model_name2)
                                os.chdir(model_name2)


                                for rep3 in range(repetitions_period):
                                        period_value = period[rep3]
                                        model_name3='analysis_{}_{}_{}_{}_{}'.format(int(Dmax*100),
int(size*100), int(angle), int(rug_value*1000), int(period_value*1000))
                                        if not os.path.isdir(model_name3):
                                                os.mkdir(model_name3)
                                        os.chdir(model_name3)


                                        model = mdb.Model(modelType=STANDARD_EXPLICIT,
name=model_name3)
                                        D, L, aspect_ratio = size, 2.0, 1
                                        x0, y0, z0 = 0.0, 0.0, 0.0
                                        generate_cylindrical_rve_ellips(D, Dmin, Dmax, rug[rep2],
period[rep3], angles[rep1], L, aspect_ratio, x0, y0, z0, mesh_size=0.07, refined_size=0.03)
                                        mat_name = 'alsi10mg'
                                        rho, E, v = 2.7e-6, 53e3, 0.33
                                        assign_material(mat_name, rho, E, v)
                                        apply_bcs(0.2)
```

```
                                    job = run_job(model_name3, 32)
                                    job.waitForCompletion()


                                    odb = openOdb(model_name3 + '.odb')
                                    max_frames = len(odb.steps['Static_1'].frames)
                                    stress_avg = np.zeros((max_frames, 1))
                                    strain_avg = np.zeros((max_frames, 1))
                                    failed_elements = np.zeros((max_frames, 1))
                                    percentage_failed_elements = np.zeros((max_frames, 1))
                                    total_volume = size ** 2 * np.pi / 4 * 2
                                    strain_limit = 0.127986583
                                    for frame_i, frame in
enumerate(odb.steps['Static_1'].frames):
                                        stressValues =
frame.fieldOutputs['S'].bulkDataBlocks[0].data[:, 1]
                                        volumeValues =
frame.fieldOutputs['EVOL'].bulkDataBlocks[0].data[:, 0]
                                        strainValues =
frame.fieldOutputs['LE'].bulkDataBlocks[0].data[:, 1]
                                        num_elements = len(strainValues)  # Numero totale di
elementi nel frame
                                        total_weighted_stress =
np.sum(np.multiply(stressValues, volumeValues))
                                        total_weighted_strain =
np.sum(np.multiply(strainValues, volumeValues))
                                        stress_avg[frame_i] = np.divide(total_weighted_stress,
total_volume)
                                        strain_avg[frame_i] = np.divide(total_weighted_strain,
total_volume)
                                        failed_elements[frame_i] =
np.count_nonzero(strainValues > strain_limit)  # Conto degli elementi con strainValues >
strain_limit
                                        # Calcolo della percentuale di elementi che superano
il valore limite rispetto al totale degli elementi per ciascun frame
                                        percentage_failed_elements[frame_i] =
(failed_elements[frame_i] / num_elements) * 100
                                        if percentage_failed_elements[frame_i] > 10:
                                            break  # Uscita dal ciclo se la soglia è stata
superata
```

```
                                    # Inserimento valori stress e strain in un 2D array
                                    data_matrix2 = np.column_stack((stress_avg[:frame_i+1],
strain_avg[:frame_i+1]))
                                    # Creazione intestazione
                                    header2 = "Stress Strain"
                                    # Salvataggio dati in file txt
                                    output_file_path2 =
'stressstrain_AR1_{}_{}_{}_{}.txt'.format(int(Dmax*100), int(angle), int(rug_value*1000),
int(period_value*1000))
                                    np.savetxt(output_file_path2, data_matrix2, header=header2,
comments='', fmt='%1.5e')
                                    # Spostamento file nella nuova cartella
                                    new_folder = os.path.join(current_directory,
output_directory, 'output_files')
                                    if not os.path.isdir(new_folder):
                                            os.mkdir(new_folder)
                                    new_file_path = os.path.join(new_folder,
os.path.basename(output_file_path2))
                                    shutil.move(output_file_path2, new_file_path)
                                    odb.close()
                                    # Eliminazione file .odb, .prt, e .sim
                                    for ext in ['.odb', '.prt', '.sim']:
                                            file_to_delete = model_name3 + ext
                                            if os.path.isfile(file_to_delete):
                                                    os.remove(file_to_delete)
                                    os.chdir('..')
                            os.chdir('..')
                    os.chdir('..')
        os.chdir('..')
# Torna alla directory iniziale
os.chdir(current_directory)
```

Bibliografia

[1] https://3dprinting.com/tips-tricks/3d-printed-lattice-structures/

[2] Giorgio De Pasquale, Antonio Coluccia, “Fatigue failure prediction in lattice structures through numerical method based on de-homogenization process”, Department of Mechanical and Aerospace Engineering, Politecnico di Torino, 2022

[3] Jacob Fish, Gregory J. Wagner and Sinan Keten, “Mesoscopic and multiscale modelling in materials”, Nature Materials vol. 20, 2021

[4] Haolin Li, Zahra Sharif Khodaei, M.H. Ferri Aliabadi, “Multiscale modelling of material degradation and failure in plain woven composites: A novel approach for reliable predictions enabled by meta-models”, Department of Aeronautics, Imperial College London, London, 2023

[5] Alessio Centola, “Machine Learning Methods to predict the Fatigue Life of Selectively Laser Melted Ti6Al4V Components”, Politecnico di Torino, 2023

[6] Erhai Hu, Ian P. Seetoh, Chang Quan Lai, “Machine learning assisted investigation of defect influence on the mechanical properties of additively manufactured architected materials”, Nanyang Technological University, Singapore, 2022

[7] Chao Wang, Yali Yang, Hao Chen, Sha Xu, Yongfang Li, Ruoping Zhang, Ming Ling, “Fatigue life prediction driven by mesoscopic defect data”, Shanghai University of Engineering Science, Shanghai, 2023